BBC MICROBASE SERIES
BBC 6502 Machine Code by Geoff Cox

This series was first published on Micronet
between April and October 1991

The BBC Micro Operating System
Part One: The moving electron writes

In the last series we reviewed the basics of machine code programming using the 6502. You will have noticed that not too many examples of programming were given. This is because there are two levels of programming on any machine, machine level and operating system level.

Operating Systems

An operating system is basically a group of routines that sit between the user and the electronics of the computer. To illustrate what an operating system does we have to turn briefly from the path of the series.

We'll imagine that you want to write the letter A on the screen of your monitor. First we have to work out the shape of the character and slice it horizontally into eight sections, one for each of eight screen scanning lines on the monitor.

Now we need to detect when the frame synchronising pulse for the monitor is sent by the computer. Next we need to count the line synchronising pulses to find the one corresponding to the start of the first line of the character.

Then we must time from this pulse to the start of the first line of the character, turn on the appropriate electron guns in the monitor and turn them off at the correct time for the end of the of the character. Finally we have a few microseconds to do it all again for the next line. That's more or less what happens fifty times a second on your monitor screen.

Mapped Screens

This makes even the easiest task very complex. We can make life easier by storing a picture or map of the screen in memory and writing the character shape to the appropriate map locations.

The map can then be scanned in synchronisation with the electron beam on the monitor. This can either be via a clever piece of electronics or a software routine.

To illustrate the BBC map, type the following program on any 6502-based BBC without screen RAM shadowing.

```
10 MODE 0
20 ?&7D00=65
```

You should see two little white dots towards the bottom right of the screen. Now two dots are not the letter A so if we want to write A we have to designthe character and write it to the map. This program does just that.

```
10 MODE 0
```

```
20 FOR A=&7D00 TO &7D07
30 READ B
40 ?A=B
50 NEXT
60 DATA &3C, &66, &66, &7E
70 DATA &66, &66, &66, &00
```

The sharp-eyed among you will have spotted something a little odd here. The screen seems to be arranged in blocks of eight bytes in this mode. Don't worry about this - it simply makes character table design simpler.

Character Tables

We can make life even easier by designing a set of characters and putting them in an area of memory where they can be looked up. In the BBC B this area is in ROM starting at &C000.

A quick diversion here. If we have to have a series of character designs in memory it helps to have a standard method of accessing them.

The standard character ordering is called ASCII. A space has ASCII number 32 and this is the first "printable" character in the set. The last is 127 (Delete). There are eight bytes in each character matrix so the design for any character is at &C000+(8*(ASCII - 32)).

This program examines the ROM character table and shows how characters are arranged.

```
10 MODE 0
20 PRINT "CHARACTER ?"
30 A$=GET$
40 PRINT A$
50 START=(8(ASC(A$)-32))+&C000
60 FIN=START+7
70 FOR BYTE=START TO FIN
80 PEEK=?BYTE
90 PROC_BIN
100 PRINT~BYTE;TAB(15);~?BYTE;TAB(20);
110 NEXT
120 END
130 DEFPROC_BIN
140 A$=""
150 FOR BIT=7 TO 0 STEP-1
160 X=(PEEK AND 2^BIT)
170 IF X>1 THEN A$=A$+"" ELSE A$=A$+"."
180 NEXT
190 ENDPROC
```

As you will often want to write characters on the screen it makes a lot of sense to have a little routine to take a character from a register and print it on the screen.

The routine will look up a character in the table and print it to the appropriate position on the screen. There will also need to be a record of the cursor position on the screen.

Routines for printing and moving a cursor on the screen will also be useful. You could, of course, write all of these routines yourself as

part of every program but as the computer manufacturer has to provide them in order to tell you the machine is working he usually leaves an access point for you to use the routines yourself.

Incidentally if you rewrite the first program but change line 10 to read MODE 7 you will see a letter A on the screen.

This is an example of clever electronics. A chip on the BBC board contains a character generator. Instead of the character design being stored in the screen map the character's ASCII value is stored. This is read by the electronics and used to generate characters directly on the screen. This allows a 40 x 80 text and block graphics screen to be generated in just 1K of RAM.

Using the Operating System

The screen handling routines and several others make up an operating system. In the BBC micro the series of routines that write a character on the screen can be accessed though a point called OSWRCH at &FFFE.

So instead of having to design characters, time lines and do all of the other things, you can output a letter A to the screen simply using:

```
LDA #65
JSR &FFEE ;OSWRCH
```

and leave the operating system to do the rest.

BBC 6502 Machine Code
Part Two: In the beginning

Last week we looked at the reasons for an operating system and how it can simplify the programmer's task. Unfortunately you will not always be able to use the operating system. There may not be a suitable routine or it may be too slow. In these cases you have to write to the device itself. So to continue our look at machine code we need to examine programming with device handling and the operating system.

This series will attempt to kill two birds with one stone by taking avery close look at Acorn's OS 1.20 used on Model Bs.

This changed little for the B+ and Master so while the routines may be in a slightly different place the basic system will be the same. The Master uses 65C02 code which has a few extra instructions so there may be some differences in the length of the code.

What you will need

A disassembler or machine code monitor that can handle 6502 or 65C02 codes. If If you don't have either you can use:

```
PRINT ~?address
```

to get the hexadecimal codes which you can then look up in the back of the BBC User Guide to get the relevant command.

Suitable Monitors are EXMON, BBC Monitor or the SYSTEM monitor. (I use "Maxim" - Ed.) If you have a single- stepping monitor like one of

these, you will be able to trace some of the routines for yourself.

For this series we will use standard 6502 mnemonics except that DB and DW will be used to show byte assignments rather than EQUS, EQUW and EQUB. This is because the disassembler I am using produces these codes and it's a lot easier to follow than convoluted BBC-type statements.

First things first

Remember that an operating system is not a program in the usual sense. Normal programs have a defined entry and exit routines. An operating system can have a large number of entry and exit points as well as interlocking routines. So to examine the operating system we need a starting point.

The 6502 regards memory as a series of 256-byte pages 0 to &FF (255). Any address can be considered to be a page number plus an offset within the page. Both figures can be represented by a single byte. So address &FF01 is on Page &FF offset 01. The concept of offsets is very useful if you ever get involved in 80n86 programming.

The BBC Manual gives a series of system entry points on page FF. Most of these are indirected through Page 2 and as we cannot guarantee what the contents of Page 2 should be (the vectors can be and are changed) these are useless as starting points. This leaves three sensible entry points.

```
6502 Vectors
FFFA
DW &0D00      ;NMI address
FFFC DW &D9CD ;RESET address
FFFE
DW &DC1C      ;IRQ address
```

The NMI address is in RAM so no joy there, but the other two look fine. The best is RESET as this is where the machine starts when it is turned on or BREAK is pressed. In the case of Model B and OS 1.20 that address is &D9CD, so what happens?

In the beginning

Reset can be effected by turning on the computer or pressing BREAK. If it is a power-up then the system VIA and processor are reset electronically.

If this is a power on situation then nothing has been set up. The first thing that happens when power is turned on is that the 6522 VIAs, the processor and the floppy disc controller are reset. This is done by means of one of three printed circuit tracks. The tracks are RSTA, RST and NOTRESET.

RSTA is only connected to the system 6522 Versatile Interface adaptor (VIA). This operates through a little resistor/capacitor circuit that only works when the power is turned on. The effect of this is that the 6522 System VIA Interrupt Enable Register (IER) bits 0 to 6 will be clear (0) only if the reset is caused by a power on condition.

If the Reset is caused by BREAK being pressed then the machine must have been on and therefore one or more of the System VIA IER bits will be set (to 1). If one or more bits are set then bit 7 of the VIA will

also be set. This is used to determine the type of Reset. So let's look at the operating system more closely.

```
D9CD
LDA
#&40  ;set NMI first instruction to RTI
D9CF
STA
&0D00 ;NMI RAM start
```

RESET is the ultimate Act of God as far as the machine is concerned. Anything could be happening so the operating system has to clean up the system as its first act.

These first instructions just make sure that if a disc is running no more information will be read or written from or to the disc. This illustrates why you shouldn't press BREAK when a disc is being accessed!

The next section sets up the stack:

```
+
D9D2
SEI  ;disable interrupts just in case
D9D3
CLD  ;clear decimal flag
D9D4
LDX
#&FF ;reset stack to where it should be
D9D6
TXS  ;(&1FF)
```

Next find out if a power-up reset or a BREAK press by examining the System VIA IER register.

```
D9D7
LDA
&FE4E ;read interrupt enable register of the system VIA
D9DA
ASL   ;shift bit 7 into carry
D9DB
PHA   ;save what's left
D9DC
BEQ
&D9E7 ;if Power up A=0 so go to D9E7 to clear memory
```

That's probably enough for this time. Don't worry! I don't intend to do a complete disassembly of the operating system in this series but we will follow through the power-on sequence to the end because a lot of interesting things happen at this time.

We'll take a look at D9E7 and the next routine in this sequence (D9DE) in the next part.

BBC 6502 Machine Code
Part Three: Cleaning up the mess

In the last part we looked at what happens when you press BREAK or switch on the machine. We'll now continue with a look at an

undocumented (at least officially) routine.

The byte at &258 can be used to contain information about what the machine should do if BREAK is pressed. FX200,n is used to set this byte. If n=2 or n=3 then the memory must be cleared. This is often used in program protection.

```
D9DE
LDA
&0258 ;else if BREAK pressed read BREAK Action flags (set by FX200,n)
D9E1
LSR

      ;divide by 2
D9E2
CMP
#&01  ;if &0258 <> 2 or 3
D9E4
BNE
&DA03 ;then Goto &DA03
D9E6
LSR

      ;divide A by 2 again (A=0 if FX200,2/3 else A=n/4
```

Pages 4-&7F are cleared by a simple loop if &258=2 or 3 or it is a power on reset. Look out for the clever way of avoiding problems on 16K machines.

```
D9E7
LDX
#&04    ;get page to start clearance from (4)
D9E9
STX
&01     ;store it in ZP 01
D9EB
STA &00 ;store A at 00
D9ED
TAY

        ;and in Y to set loop counter

        ;LOOP STARTS
D9EE
STA
(&00),Y ;clear RAM
D9F0
CMP
&01     ;until page address (in &01) =0
D9F2
BEQ
&D9FD
;
D9F4
INY

        ;increment pointer
D9F5
BNE
&D9EE   ;if not zero loop round again
```

```
D9F7
INY     ;else increment again (Y=1) this avoids overwriting the RTI
        ;instruction at &D00 D9F8 INX

        ;increment X
D9F9
INC
&01     ;increment &01
D9FB
BPL
&D9EE   ;loop until Page (in 01)=&80 then exit
```

Note that RAM addressing for 16K loops around to &4000=&00 hence the checking of &01 for 00. This avoids overwriting zero page on BREAK which would cause the machine to crash!

```
D9FD
STX
&028E   ;writes marker for available RAM 40 =16K,80=32
DA00
STX
&0284   ;write soft key consistency flag
```

This routine shows the basic structure of a loop. Those of you who program in BASIC will recognise it as a very simple structure:

```
10 A=A+1
20 IF A<20 GOTO 10
```

The loop uses zero page addressing with the target address in 00 and 01 (Page) and the index in Y.

The loop is exited when the value in 01 becomes negative. Remember that all values between 0 and &7F are considered to be positive, so the BPL instruction can be used to exit the loop at page &80, the first negative number. This is the first of the useful loop techniques we'll see in this series.

Notice that the first byte of each page is left unchanged. This is useful if you want information to survive a BREAK of this type. This clearing of memory is not normally carried out.

Next week we'll have a look at the normal RESET path.

BBC 6502 Machine Code
Part Four: Cleaning up even more mess

As we saw last week, a normal warm reset avoids the memory clearance and proceeds to set up the System VIA.

```
DA03 LDX #&0F  ;set PORT B data direction register to output on bits
               ;0-3 and input bits 4-7
DA05 STX &FE42 ;
```

The next bit is a little more complicated and is intimately bound up with hardware. The function is to set up the addressable latch IC 32 for peripherals via PORT B.

The latch value is written by writing the value to &FE40 bits 0 to 2 and either a 1 or 0 to bit 3.

Writing the value + 8 therefore writes a 1 to the latched address, otherwise a 0 is written.

Value +	Peripheral	Effect 0	8
0	Sound chip	Enabled	Disabled
	Speech Chip		
1	(RS)	Low	High
2	(WS)	Low	High
2	(WS)	Low	High
3	Keyboard		
	Write	Disabled	Enabled
4	C0 address		
	modifier	Low	High
5	C1 address		
	modifier	Low	High
6	Caps LED	On	Off
7	Shift LED	On	Off

C0 and C1 are involved with hardware scroll screen address.

```
                    ;X=&F on entry
DA08   DEX          ;loop start
DA09   STX   &FE40  ;Write latch IC32
DA0C   CPX   #&09   ;Is it 9?
DA0E   BCS   &DA08  ;If not go back and do it again
                    ;X=8 at this point
                    ;Caps Lock On, SHIFT Lock undetermined
                    ;Keyboard Autoscan on
                    ;Sound disabled (may still sound)
```

Next the keyboard is scanned to determine the values of the keyboard links and whether a Ctrl-Break has been performed.

Remember that although we have spent a lot of time reading this, we are probably less than 200 microseconds after BREAK was pressed.

The check for Ctrl-Break is effectively looking for simultaneous keypresses.

```
DA10   INX          ;X=9
DA11   TXA          ;A=X
DA12   JSR   &F02A  ;Interrogate keyboard
DA15   CPX   #&80   ;for keyboard links 9-2 and CTRL key (1)
DA17   ROR   &FC    ;rotate MSB into bit 7 of &FC

DA19   TAX          ;Get back value of X for loop
DA1A   DEX          ;Decrement it
DA1B   BNE   &DA11  ;and if >0 do loop again
                    ;On exit if Carry set link 3 is made
                    ;link 2 = bit 0 of &FC and so on
                    ;If CTRL pressed bit 7 of &FC=1 X=0
DA1D   STX   &028D  ;Clear last BREAK flag
DA20   ROL   &FC    ;CTRL is now in carry &FC is keyboard links
DA22   JSR   &EEEB  ;Set LEDs
                    ;Carry set on entry is in bit 7 of A on exit
DA25   ROR          ;Get carry back into carry flag
```

To review what the operating system has done so far, about 400 microseconds after a BREAK press or about 2 milliseconds from a power on. Memory may have been cleared, NMIs have been short circuited, IRQs disabled. The keyboard has been scanned for made links and for Ctrl being pressed.

We have also located two important and undocumented subroutines: &F02A toscan the keyboard and &EEEB to set the keyboard LEDs.

The F02A routine scans for the key whose code is in X being pressed:

```
F02A   LDY   #&03  ;Stop Auto scan
F02C   STY   &FE40 ;by writing to system VIA
F02F   LDY   #&7F  ;Set bits 0 to 6 of port A to input on bit 7.
                   ;Output on bits 0 to 6
F031   STY   &FE43 ;
F034   STX   &FE4F ;Write X to Port A system VIA (key to check)
F037   LDX   &FE4F ;Read back &80 if key pressed (M set)
F03A   RTS         ;And return
```

The routine at &EEEB switches on the selected keyboard lights.

```
EEEB   PHP          ;Save flags
EEEC   LDA   &025A  ;Read keyboard status
                    ;Bit 7=1 shift enabled
                    ;Bit 6=1 control pressed
                    ;Bit 5 =0 shift lock
                    ;Bit 4 =0 Caps lock
                    ;Bit 3 =1 shift pressed
EEEF   LSR          ;Shift Caps bit into bit 3
EEF0   AND   #&18   ;Mask out all but 4 and 3
EEF2   ORA   #&06   ;Returns 6 if caps lock OFF &E if on.
                    ;Remember add 8 to the value for the addressable
                    ;latch to send a 1.
EEF4   STA   &FE40  ;Turn on or off caps light if required
EEF7   LSR          ;Bring shift bit into bit 3
EEF8   ORA   #&07   ;
EEFA   STA   &FE40  ;Turn on or off shift lock light
EEFD   JSR   &F12E  ;Set keyboard counter
EF00   PLA          ;Get back flags into A
EF01   RTS          ;Return
```

In this part we've had a look at subroutines using JSR and RTS, the machine code equivalent of GOSUB, PROC or FN. Subroutines are often used in machine code to perform such frequently needed functions as scanning a keyboard or turning on and off lights.

We've also discovered that the byte at &25A contains the keyboard status. Try changing it for yourself. You can therefore use OR and AND to set the shift and Caps lock status of the machine for a particular program.

Next week we'll examine setting up the default vector table in memory.

BBC 6502 Machine Code
Part Five: Vectors Victor

The next stage is to set up the vectors on page 2.

```
DA26   LDX   #&9C  ;
```

```
DA28   LDY   #&8D  ;
DA2A   PLA         ;Get back A from &D9DB DA2B
BEQ    &DA36       ;If A=0 power up reset so go to DA36 with X=&9C
                   ;Y=&8D
DA2D   LDY   #&7E  ;else let Y=&7E
DA2F   BCC   &DA42 ;and if not CTRL- BREAK go to DA42 for a WARM RESET
DA31   LDY   #&87  ;else Y=&87 COLD RESET
DA33   INC   &028D ;&28D=1
DA36   INC   &028D ;&28D=&28D+1
DA39   LDA   &FC   ;Get keyboard links set
DA3B   EOR   #&FF  ;Invert
DA3D   STA   &028F ;and store at &28F
DA40   LDX   #&90  ;X=&90
```

What we have done is to set up the high water marks for the reset of vectors.

```
&28D=0 Warm reset, X=&9C, Y=&7E
&28D=1 Power up  , X=&90, Y=&8D
&28D=2 Cold reset, X=&9C, Y=&87
```

```
DA42   LDA   #&00    ;A=0
DA44   CPX   #&CE    ;zero &200+X to &2CD
DA46   BCC   &DA4A   ;
DA48   LDA   #&FF    ;then set &2CE to &2FF to &FF
DA4A   STA   &0200,X ;
DA4D   INX           ;
DA4E   BNE   &DA44   ;
                     ;A=&FF X=0
```

This is another IF-GOTO loop, but in this case it is a double function loop. The test at DA44 to DA46 means that A is 0 only for values of X between the high water mark and &CD. Above this value A is set to &FF by the instruction at &DA48. This saves a few bytes of space, essential when writing a tightly-filled ROM.

The next instructions set up the printer port. The only reason for doing this now is to save two bytes. A must be &FF at this point so it is used to set up the User VIA for outputs as the printer port.

```
DA50   STA   &FE63   ;Set port A of user VIA to all outputs (printer out)
DA53   TXA           ;A=0 DA54   LDX   #&E2    ;X=&E2

START OF LOOP
DA56   STA   &00,X   ;set zero page addresses &E2 to &FF to zero
DA58   INX           ;
DA59   BNE   &DA56   ;X=0
```

Now set up the vectors in page 2 from the table at &D940:

```
DA5B   LDA   &D93F,Y ;copy data from &D93F+Y
DA5E   STA   &01FF,Y ;to &1FF+Y
DA61   DEY           ;until
DA62   BNE   &DA5B   ;1FF+Y=&200
```

Note that this is a decrementing loop which, for loops ending when an index register reaches zero, is faster and shorter because no compare is needed. More space saved!

Now the RS423 port is set up via a subroutine affecting the ACIA.

(Asynchronous Communications Interface Adaptor)

```
DA64   LDA   #&62    ;A=&62
DA66   STA   &ED     ;store in &ED
DA68   JSR   &FB0A   ;set up ACIA ;X=0
```

Now Acorn clears the interrupt and enable registers of both VIAs.

```
DA6B   LDA   #&7F    ;bit 7 is 0!
DA6D   INX           ;
DA6E   STA   &FE4D,X ;
DA71   STA   &FE6D,X ;
DA74   DEX           ;
DA75   BPL   &DA6E   ;
                     ;This loop only has two passes as X=0 on entry.
DA77   CLI           ;Briefly allow interrupts to clear anything
                     ;pending
DA78   SEI           ;Disallow again NB: all VIA IRQs are disabled
DA79   BIT   &FC     ;If bit 6=1 then JSR &F055 as there must be a
                     ;hardware interrupt!
DA7B   BVC   &DA80   ;else DA80
DA7D   JSR   &F055   ;
```

What have we here? Another undocumented routine. If bit 6 of &FC is set there must have been a hardware interrupt when the SEI occurred.

From the circuit diagram the only place that this IRQ could have come from is the 1MHz bus - let's have a look at the routine at &F055.

```
F055   JMP   (&FDFE) ;Jim paged entry vector
```

So we jump to some piece of hardware on the 1MHz bus. This would probably be a ROM which would take over the system at power on and Break. This has some very interesting applications. It was designed by Acorn to provide a crude Econet facility to allow a batch of machines to be functionally tested without the need to install a full Econet kit.

Next week we shall examine the VIA bus.

BBC 6502 Machine Code
Part Six: The VI bus

The next interesting routine we find in the BBC operating system is the one that sets up the system VIA interrupts. It is located at &DA80. Refer to the manual for the meanings of Sheila addresses.

```
DA80   LDX   #&F2  ;Enable interrupts 1,4,5,6 of system VIA
DA82   STX   &FE4E ;
                   ;0 Keyboard enabled as needed
                   ;1 Frame sync pulse
                   ;4 End of A/D conversion
                   ;5 T2 counter (for speech)
                   ;6 T1 counter (10 mSec intervals)

DA85   LDX   #&04  ;set system VIA PCR
DA87   STX   &FE4C ;
                   ;CA1 Interrupt on negative edge (Frame sync)
                   ;CA2 Handshake output for keyboard
                   ;CB1 Interrupt on negative edge (end of conversion)
```

```
                    ;CB2 Negative edge (Light pen strobe)
DA8A   LDA   #&60   ;Set system VIA ACR
DA8C   STA   &FE4B  ;
                    ;Disable latching
                    ;Disable shift register
                    ;T1 counter continuous interrupts
                    ;T2 counter timed interrupt

DA8F   LDA   #&0E   ;Set system VIA T1 counter (low)
DA91   STA   &FE46  ;
                    ;This becomes effective when T1 hi set
DA94   STA   &FE6C  ;Set user VIA PCR
                    ;CA1 interrupt on -ve edge (Printer Acknowledge)
DA80   LDX   #&F2   ;enable interrupts
                    ;CA2 High output (printer strobe)
                    ;CB1 Interrupt on -ve edge (user port)
                    ;CB2 Negative edge (user port)
DA97   STA   &FEC0  ;Set up A/D converter Bits 0 and 1 determine
                    ;channel selected
                    ;If Bit 3=0 it is set for an 8-bit conversion.
                    ;If bit 3=1 12-bit conversion.
```

Now although the machine now knows how much RAM it has it still doesn't know if it's a Model A or Model B, so it does not know if a user VIA is present at &FE60-FE6F.

The next routine tests for the presence of a user VIA. The system timers are then set up to interrupt every 10mSec. Sound channels are cleared and the serial ULA is set up. Then the function keys are reset.

Now we need a catalogue of sideways ROMS. This is not a catalogue in the conventional sense as the ROM title is always at the same place in the ROM itself and can be read from there. It is a catalogue of the ROM types and positions.

There is a ROM latch at &FE30. Writing a number between 0 and 15 to this switches the corresponding ROM into the area between &8000 and &BFFF. A short subroutine does this and maintains a copy of the current ROM in zero page at location &F4.

```
                    ;on entry X=required ROM number
DC16   STX   &F4    ;RAM copy of ROM latch
DC18   STX   &FE30  ;Write to ROM latch
DC1B   RTS          ;and return
```

You should use this subroutine if you want to switch ROMs. Now we can look at the ROM cataloguing routines;

A ROM is considered to be valid if it contains a string identical to astring at location &DF0C in the Operating System ROM.

```
DF0C   DB    ')C('   ;
DF0F   DB    0       ;
```

The location of this string is pointed to by an offset byte located at &8007.

```
;X=0 on entry
DABD   JSR   &DC16   ;Set up ROM latch and RAM copy to X
```

```
DAC0   LDX   #&03    ;Set X to point to offset in table
DA80   LDX   #&F2    ;Enable interrupts
DAC2   LDY   &8007   ;Get copyright offset from ROM
DAC5   LDA   &8000,Y ;Get first byte
DAC8   CMP   &DF0C,X ;Compare it with table byte
DACB   BNE   &DAFB   ;If not the same then goto DAFB
DACD   INY           ;Point to next byte
DACE   DEX ;(s)
DACF   BPL   &DAC5   ;and if still +ve go back to check next byte.
                     ;This point is reached if 4 bytes indicate
                     ;valid ROM
```

Next the first 1K of each ROM is checked against higher priority ROMs to ensure that there are no matches. If a match is found, the lower priority ROM is ignored.

A ROM type byte is located at &8006. A catalogue of these bytes is held at &2A1-&2B0. If bit 7 of this byte is 0 then the ROM is BASIC. The position of this ROM is stored at &24B.

Now the ROMs are catalogued it is time to set up the speech system and screen. More about that next week.

BBC 6502 Machine Code
Part Seven: Talk to me

The operating system start-up routines next checks the SPEECH system. At this point the X register is set to 16 (&10) by previous routines.

This is one of the reasons why this routine is inserted here. Setting X to the required value would use two more bytes. This is not much space but it can make the difference between all of the OS fitting into a single ROM and a complete hardware or software redesign.

```
DB11  BIT  &FE40  ;If bit 7 low then we have speech system fitted
DB14  BMI  &DB27  ;else goto DB27 for screen set up routine.
DB16  DEC  &027B  ;(027B)=&FF a RAM flag that indicates that a speech
                  ;chip is present.
DB19  LDY  #&FF   ;Y=&FF
DB1B  JSR  &EE7F  ;Initialise speech generator
DB1E  DEX         ;via this
DB1F  BNE  &DB19  ;loop
```

Now X = 0 so:

```
DB21  STX  &FE48  ;Set T2 timer for speech
DB24  STX  &FE49  ;
```

Screen set-up

X=0 on entry to this routine which gets the default screen mode and then goes off to the screen setup routine.

```
DB27  LDA  &028F  ;Get back start up options (mode)
DB2A  JSR  &C300  ;then jump to initialise screen.
```

One of the things that I wondered when I got a BBC was how the RESET key could possibly act as a soft key. As we all know BREAK acts as soft key 10. But the keyboard buffer is cleared by the Reset. Tucked away is the five-byte routine that makes the BREAK key act as soft

key 10.

Soft keys work by inserting a byte greater than 127 into the keyboard buffer. &CA is the code for key 10.

```
DB2D  LDY  #&CA   ;Y=&CA
DB2F  JSR  &E4F1  ;to enter this value in the keyboard buffer
```

Simple isn't it? You can use the routine yourself although further investigation will show that E4F1 is part of an OSbyte call. Remember that the keyboard buffer is buffer 0.

```
E4F1  LDX  #&00  ;X=0 keyboard buffer
```

```
************************************
*                                  *
*   OSBYTE 153 Put byte in input   *
*    Buffer checking for ESCAPE    *
*                                  *
************************************
```

On entry X = buffer number which is either 0 or 1. If it's 0 then the keyboard buffer is selected. If it's 1 then it is the RS423 buffer.

Notice that the JSR to EF41 ensures that ONLY the keyboard buffer can be selected. Once again we are looking at coding economy, in this case with a specific keyboard buffer entry routine. Y contains the 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
```

This routine will normally be accessed by assembly language programmers by OSbyte 138 which calls EF43.

BBC 6502 Machine Code
Part Eight: Breaker Break

One of the 'secret' features of the BBC Micro OS 1.20 when it was arrived was the BREAK intercept. This is a useful method of taking over the machine and is sometimes used by ROM software.

There are two entry points, entered with the carry flag reset to 0 and

set to 1 respectively. The first call comes before sideways ROM calls.

Enter BREAK intercept with Carry Clear

```
DB32  JSR  &EAD9  ;check to see if BOOT address is set up if so
                  ;JMP to it
```

The address &287 is written by OSbyte 247 and the jump addresses in &288 and &289 by OSbytes 248 and 249. The machine code for JMP is &4C.

```
EAD9  LDA  &0287  ;get BREAK vector code
EADC  EOR  #&4C   ;produces 0 if JP (4C) not in &287
EADE  BNE  &EAF3  ;if not goto EAF3
EAE0  JMP  &0287  ;else jump to use
BREAK code
EAF3  RTS         ;Return
```

The RTS at the end of another routine is used because it saves code.

Frequently you will find machine code routines where a lot of branches go to a single RTS for just this reason. If you are writing your own code remember that the RTS must be within range of the branch. One of the most common assembler errors is a branch out of range that in turn causes more errors when you add an extra RTS.

Obviously at this point the machine could be totally in your control. You can return control to the OS with an RTS or just continue on your merry way.

Remember that the sideways ROMs don't have any workspace yet and you can't really run BASIC or any other language as the workspace will not exist. But, assuming that you don't want to do any of this, let's go back to the OS routines after testing for BREAK intercept.

```
DB35  JSR  &F140  ;set up cassette options
DB38  LDA  #&81   ;test for tube to FIFO buffer 1
DB3A  STA  &FEE0  ;
DB3D  LDA  &FEE0  ;
DB40  ROR         ;put bit 0 into carry
DB41  BCC  &DB4D  ;if no tube then DB4D
DB43  LDX  #&FF   ;else
DB45  JSR  &F168  ;issue ROM service call &FF to initialise TUBE system
DB48  BNE  &DB4D  ;if not 0 on exit (tube not initialised) DB4D
DB4A  DEC  &027A  ;else set tube flag to show its active
```

Now the Tube is flagged as active, or not as the case may be. We continue next week, with the setup routines for the sideways ROMs.

BBC 6502 Machine Code
Part Nine: A ROM with a view

Now we nearly have a working system, we are, perhaps, 400 milliseconds into the Power up routine. Now is the time to set up all of those nice sideways ROMs we catalogued earlier.

First we set up workspace and hence the value of BASIC's PAGE variable. The call to ROMs is made via F168. This is available to the programmer as OSBYTE 143.

A ROM can have a number between 0 and 15 and will have two entry

points - a Service entry at &8003 and a Language entry at &8000. If the ROM does not contain language code it will not have a language entry.

ROMs are paged into main memory by writing the ROM number to a latch at &FE30. Hardware could be arranged to allow 256 ROMs although the operating system does not support this.

The Break Intercept code could be used to make drastic hardware modifications like this.

```
************************************
*                                  *
* OSBYTE 143                       *
* Pass service commands            *
* to sideways ROMs                 *
*                                  *
************************************
                  ;on entry X=command number
F168  LDA  &F4    ;get current ROM number
F16A  PHA         ;store it
F16B  TXA         ;command in A
F16C  LDX  #&0F   ;set X=15
```

The next bit of code is a countdown loop to send the command code to each enabled ROM in turn. The Map at &2A1 is used to decide which ROMs are active. Note the use of a countdown loop. This gives code economy and explains why the highest ROM number has priority.

```
F16E  INC  &02A1,X ;read bit 7 on ROM map (no ROM has type 254 &FE)
F171  DEC  &02A1,X ;
F174  BPL  &F183   ;if not set (+ve result)
F176  STX  &F4     ;else store ROM number in &F4
F178  STX  &FE30   ;switch in paged ROM
F17B  JSR  &8003   ;and jump to service entry
F17E  TAX          ;on exit put A in X
F17F  BEQ  &F186   ;if 0 (command recognised by ROM) reset ROMs & exit
F181  LDX  &F4     ;else point to next lower ROM
F183  DEX          ;
F184  BPL  &F16E   ;and go round loop again
F186  PLA          ;get back original ROM number
F187  STA  &F4     ;store it in RAM copy
F189  STA  &FE30   ;select original page
F18C  TXA          ;put X back in A
F18D  RTS          ;and return
```

Couldn't be easier! So we can now return to the main body of the routine.

```
DB4D  LDY  #&0E    ;set current value of PAGE
DB4F  LDX  #&01    ;issue call to claim absolute workspace
DB51  JSR  &F168   ;via F168
DB54  LDX  #&02    ;send private workspace claim call
DB56  JSR  &F168   ;via F168
```

OSHWM is OS High Water Mark. The highest address used by the operating system.

```
DB59  STY  &0243   ;set primary OSHWM DB5C
STY  &0244   ;set current OSHWM
```

```
DB5F LDX   #&FE    ;issue call for Tube to explode character set etc.
DB61  LDY  &027A   ;Y=FF if tube present else Y=0
DB64  JSR  &F168   ;and make call via F168
```

We now have the machine set up to enter a language, all the filing systems have been set up and the sideways ROMs activated.

Next week we finally start the screen messages.

BBC 6502 Machine Code
Part Ten: Stringing it along

The next routine shows why the Machine start up message is not always seen on third-party kit.

```
DB67  AND  &0267  ;if A=&FE and bit 7 of 0267 is set then continue
DB6A  BPL &DB87   ;else ignore start up message
DB6C  LDY  #&02   ;output to screen
DB6E  JSR  &DEA9  ;'BBC Computer ' message
```

Looking at the routine in DE9A we find a very useful string printing routine. Remember that Y = 2 on entry.

```
DEA9  LDA  #&C3   ;point to start &C300
DEAB  STA  &FE    ;store it
DEAD  LDA #&00   ;point to lo byte
DEAF  STA  &FD    ;store it and start loop with Y=2
DEB1  INY         ;print character in string
DEB2  LDA (&FD),Y ;pointed to by &FD/E +Y
DEB4  JSR  OSASCI ;print it expanding Carriage returns
DEB7  TAX ;store A in X
DEB8  BNE  &DEB1  ;and loop again if not =0
DEBA  RTS         ;else exit
```

Here is the string delimited by BRK. The code for BRK is 00. Y is 3 when the first character is read so its address is &C303.

```
C303  DB  13      ;Carriage Return
C304  DB 'BBC Computer '
C311  BRK
```

Notice that the routine uses TAX to set the zero flag which marks the end of the string. This is a useful tip.

The next part of the Operating system deals with printing correct messages on the screen.

```
DB71  LDA  &028D  ;0=warm reset, If a cold reset continue
DB74  BEQ  &DB82  ;
DB76  LDY  #&16   ;by checking length of RAM
DB78  BIT  &028E  ;
DB7B  BMI  &DB7F  ;and either
DB7D  LDY  #&11   ;
DB7F  JSR  &DEA9  ;finishing message with '16K' or '32K'
DB82  LDY  #&1B   ;and two new lines
DB84  JSR  &DEA9  ;
```

Notice that Y is used to pick the appropriate message.

```
C312  DB  '16K'
```

```
C315  DB  7       ;Bell
C316  BRK
C317  DB  '32K'
C31A  DB  7       ;Bell
C31B  BRK
C31C  DB  08,0D,0D
```

Notice the BBC Beep at this point indicates that nearly all set up procedures have been finished.

The hum is generated by the Sound channel which is reset as part of the start routine. Hence the HUM-BEEP start up. If the machine does not start properly the sound signals give a strong clue to the nature of the problem. Having got this far the OS gives us another chance to take control.

Enter BREAK INTERCEPT ROUTINE WITH CARRY SET (call 1)

```
DB87  SEC         ;
DB88  JSR  &EAD9  ;look for break intercept jump
                  ;SEE EARLIER PART
```

Next we set up the keyboard lights

```
DB8B  JSR  &E9D9  ;set up LEDs in accordance with keyboard status
```

This is another 'undocumented' OSBYTE call.

```
************************************
*                                  *
* OSBYTE &76 (118)                 *
* SET LEDs to Keyboard Status      *
*                                  *
************************************
;osbyte entry with carry set
E9D9  PHP ;PUSH P
E9DA  SEI         ;DISABLE INTERRUPTS
E9DB  LDA  #&40   ;switch on CAPS and SHIFT lock lights
E9DD  JSR  &E9EA  ;via subroutine
E9E0  BMI  &E9E7  ;if ESCAPE exists (M set) E9E7
E9E2  CLC         ;else clear V and C
E9E3  CLV         ;before calling main keyboard routine to
E9E4  JSR  &F068  ;switch on lights as required
E9E7  PLP ;get back flags
E9E8  ROL         ;and rotate carry into bit 0
E9E9  RTS ;Return to calling routine
                  ;
* Turn on keyboard lights and
* Test Escape flag
                  ;
E9EA  BCC  &E9F5  ;if carry clear
E9EC  LDY  #&07   ;switch on shift lock light
E9EE  STY  &FE40  ;
E9F1  DEY         ;Y=6
E9F2  STY  &FE40  ;switch on Caps lock light
E9F5  BIT  &FF    ;set minus flag if bit 7 of &00FF is set indicating
E9F7  RTS         ;that ESCAPE condition exists, then return
```

The Keyboard routine continues via the KEYV. This is a little long to include here so we'll leave it until a later part. So back to the

Start up routine next week with the cassette system.

BBC 6502 Machine Code
Part Eleven: Language!

Having got the keyboard nicely set up the machine proceeds to initialise a filing system and run a !BOOT file if one exists. The start up options are already read from the keyboard links.

```
DB8E  PHP         ;save flags
DB8F  PLA         ;and get back in A
DB90  LSR ;zero bits 4-7 and bits 0-2 bit 4 which was bit 7
DB91  LSR         ;may be set
DB92  LSR         ;
DB93  LSR         ;
DB94  EOR  &028F  ;EOR with start up options which may or may not
DB97  AND  #&08   ;invert bit 4
DB99  TAY  ;Y=A
DB9A  LDX  #&03   ;make initialisation call if Y=0 on entry
DB9C  JSR  &F168 ;RUN, EXEC or LOAD !BOOT file from a filing system.
DB9F  BEQ  &DBBE  ;if a ROM accepts this call then
DBBE
DBA1  TYA         ;else put Y in A
DBA2  BNE  &DBB8  ;if Y<>0 DBB8
DBA4  LDA  #&8D   ;else set up standard cassette baud rates
DBA6  JSR  &F135  ;via &F135 which is OSBYTE 140.
DBA9  LDX  #&D2   ;
DBAB  LDY  #&EA   ;
DBAD  DEC  &0267  ;decrement ignore start up message flag
DBB0  JSR  OSCLI  ;and execute /!BOOT
DBB3  INC  &0267  ;restore start up message flag
DBB6  BNE  &DBBE  ;if not zero then DBBE
DBB8  LDA  #&00   ;else A=0
DBBA  TAX  ;X=0
DBBB  JSR  &F137  ;set tape speed via OSBYTE 140.
```

We now have an active filing system. The next job is to preserve the current language on soft RESET.

```
DBBE  LDA  &028D  ;get last RESET Type
DBC1  BNE  &DBC8  ;if not soft reset
DBC8
DBC3  LDX  &028C  ;else get current language ROM address
DBC6  BPL  &DBE6  ;if +ve (language available) then skip search
                  ;routine
```

For a cold break we search for the language with the highest priority.

```
DBC8  LDX  #&0F   ;set pointer to highest available ROM
DBCA  LDA &02A1,X ;get ROM type from map
DBCD  ROL         ;put hi-bit into carry, bit 6 into bit 7
DBCE  BMI  &DBE6  ;if bit 7 set then ROM has a language entry so DBE6
DBD0  DEX         ;else search for language until X=&ff
```

Check for Tube if no language found.

```
DBD1  BPL  &DBCA  ;check if tube present
DBD3  LDA  #&00   ;if bit 7 of tube flag is set BMI succeeds
DBD5  BIT  &027A  ;and TUBE is connected else
DBD8  BMI &DC08  ;make error
```

No language error

```
DBDA  BRK            ;
DBDB  DB   &F9       ;error number
DBDC  DB 'Language?' ;message
DBE5  BRK            ;
```

This might seem odd as BRK is handled by the current language BRK handler,but we don't have a language! We need to investigate further in another part.

```
DBE6  CLC         ;
```

OSBYTE 142 enter Language ROM at &8000 X=ROM number. Carry is set if this is an OSBYTE call and clear if this is an initialisation routine.

```
DBE7  PHP         ;save flags
DBE8  STX  &028C  ;put X in current ROM page
DBEB  JSR  &DC16  ;select that ROM
DBEE  LDA  #&80   ;A=128
DBF0  LDY  #&08   ;Y=8
DBF2  JSR  &DEAB  ;display text string held in ROM at &8008,Y
DBF5  STY  &FD    ;save Y on exit (end of language string)
DBF7  JSR  OSNEWL ;two line feeds
DBFA  JSR  OSNEWL ;are output
DBFD  PLP         ;then get back flags
DBFE  LDA  #&01   ;A=1 required for language entry
DC00  BIT  &027A  ;check if tube exists
DC03  BMI  &DC08  ;and goto DC08 if it does
DC05  JMP  &8000  ;else enter language at &8000
```

TUBE FOUND enter tube software

```
DC08  JMP  &0400  ;enter tube environment
```

The Tube initialisation would have read the language across to the TUBE usually but it could be loaded by a !BOOT file from the filing system initialisation.

The operating system now stops general control of the system and hands this to the language which looks after command lines etc. The OS however still handles the screen, keyboard and much else.

Notice how every possible eventuality was taken into account during the initialisation routine. This is one of the things that made the Beeb a very powerful machine.

Next week we'll have a look at the Interrupt code.

BBC 6502 Machine Code
Part Twelve: Pardon me!

We finished the last part at the point where the operating systems power up routine handed over control to the language. We'll write our own language later in the series but for now let's dive into another entry point.

When the processor's RQ pin (4) goeslow (0V) the processor finishes off the current instruction and then goes off to run some microcode of its own. This checks that the RDY (2) pin is high and that the

interrupt flag in the status register is 0 (not set). If it is set the interrupt is ignored and the processor goes to the next instruction. This continues when the IRQ pin is low.

If the flag is clear then the processor stores the program counter and status register on the stack and sets the interrupt flag. The 6502 then gets the address stored in &FFFE and &FFFF and executes this instruction next.

If a BRK instruction is found in executing code then the processor performs exactly the same actions except that it does not check the status register for the interrupt flag, it does set a flag in the status register, the BRK flag.

The main entry point for IRQ (and BRK) for OS 1.20 is &DC51.

MAIN IRQ Entry point

```
;ON ENTRY STACK contains STATUS REGISTER,PCH,PCL
DC1C  STA  &FC    ;save A
DC1E  PLA         ;get back status (flags)
DC1F  PHA         ;and save again
DC20  AND  #&10   ;check if BRK flag set
DC22  BNE  &DC27  ;if so goto DC27
DC24  JMP (&0204) ;else JUMP through IRQ1V
```

That's pretty straightforward so far. As you can see IRQ1V allows you to put your own hardware at a higher priority than anything else in the machine.

You can also write your own hardware interrupt handler if you wish. This is the flexibility that made the BBC machine so remarkably successful among knowledgeable users.

Let's look at the BRK handler now.

```
* BRK handling routine *
DC27  TXA         ;save X on stack
DC28  PHA         ;
DC29  TSX         ;get status pointer
DC2A  LDA &0103,X ;get Program Counter low byte
DC2D  CLD         ;
DC2E  SEC         ;set carry
DC2F  SBC  #&01 ;subtract 2 (1+carry)
DC31  STA  &FD    ;and store it in &FD
DC33  LDA &0104,X ;get hi byte
DC36  SBC  #&00   ;subtract 1 if necessary
DC38  STA  &FE    ;and store in &FE
DC3A  LDA  &F4    ;get currently active ROM
DC3C  STA  &024A  ;and store it in &24A
DC3F  STX  &F0    ;store stack pointer in &F0
DC41  LDX  #&06   ;and issue ROM service call 6
DC43  JSR  &F168  ;(User BRK) to ROMs
                  ;now &FD/E points to byte after BRK
                  ;ROMS may use BRK for their own purposes
                  ;and many do!
```

It's interesting to see what happens with the ROM handler. This is also an entry point for OSBYTE 143 so you can use this in your own code.

```
* OSBYTE 143 *
*Pass service commands to sideways ROMs *
                  ;on entry X=command number
F168  LDA &F4     ;get current ROM number
F16A  PHA         ;store it
F16B  TXA         ;command in A
F16C  LDX  #&0F   ;set X=15
                  ;send commands loop
F16E  INC &02A1,X ;read bit 7 on ROM map (no ROM has ;type 2)
4 &FE)
F171  DEC &02A1,X ;
F174  BPL &F183   ;if not set (+ve result)
F176  STX  &F4    ;else store ROM number in &F4
F178  STX  &FE30  ;switch in paged ROM
F17B  JSR  &8003  ;and jump to service entry
F17E  TAX         ;on exit put A in X
F17F  BEQ  &F186  ;if 0 (command recognised by ROM)
                  ;reset ROMs & exit
F181  LDX  &F4    ;else point to next lower ROM
F183  DEX         ;
F184  BPL  &F16E  ;and go round loop again
F186  PLA         ;get back original ROM number
F187  STA  &F4    ;store it in RAM copy
F189  STA  &FE30  ;select original page
F18C  TXA         ;put X back in A
F18D  RTS         ;and return
```

Useful little routine that. So back to the BRK handler.

```
DC46  LDX  &028C  ;get current language
DC49  JSR  &DC16  ;and activate it
DC4C  PLA         ;get back original value of X
DC4D  TAX         ;
DC4E  LDA  &FC    ;get back original value of A
DC50  CLI         ;allow interrupts
DC51  JMP (&0202) ;and JUMP via BRKV (normally into current language)
```

Next week we'll carry on by taking a look at the BRK handler.

BBC 6502 Machine Code
Part Thirteen: Give us a BRK

BRK is usually handled by the default language (or by a Sideways ROM). However, it may be that you are running a machine code program before a current language is set up or perhaps your language doesn't handle BRK (it should but you never know).

That's when a default BRK handler takes over.

```
* DEFAULT BRK HANDLER *

DC54 LDY #&00  ;Y=0 to point to byte after BRK
DC56 JSR &DEB1 ;print message
```

Let's have a look at the print routine. Remember that the error-handling layout is:

```
BRK
Error Number (1 byte)
```

Message
BRK

Y plus the address in &FD &FE points to the error message on entry.

```
DEB1 INY          ;point to first ;character in string
DEB2 LDA (&FD),Y
DEB4 JSR OSASCI   ;print it
                  ;expanding
                  ;Carriage
                  ;returns
DEB7 TAX          ;store A in X to change flags
DEB8 BNE &DEB1    ;and loop again if not =0
DEBA RTS          ;else exit
```

A standard print routine, nothing out of the ordinary but nice and compact.

You can use this in your own print routines by changing the zero page values. Back to the default BRK handler and an interesting bit of code.

```
DC59 LDA &0267 ;if BIT 0 set and DISK EXEC error
DC5C ROR       ;occurs
DC5D BCS &DC5D ;hang up machine!
```

Nasty! But the machine has to be in a pretty unusual configuration for this to happen. Mind you, setting 0267 then doing a JSR to DC59 would confuse the average user.

```
DC5F JSR OSNEWL ;else print two newlines
DC62 JSR OSNEWL ;
DC65 JMP &DBB8  ;and set tape speed before entering the current
                ;language
DBB8 LDA #&00   ;else A=0
DBBA TAX        ;X=0
DBBB JSR &F137  ;set tape speed via OSBYTE 141.
```

There's the end of the BRK handling code. As I said before this is generally handled by the default language but you can arrange for your own code or a Sideways ROM to handle it.

Next week we'll return to the interrupt system with a look at the default entry point for IRQ1.

BBC 6502 Machine Code
Part Fourteen: The story so far...

We left the interrupt-handling routine just after it had gone off to the IRQ1V vector. If you don't change the vector the code continues from DC93.

One very important thing to remember about an interrupt-driven machine like the BBC is that the interrupt flag is not set for too long. If it is the machine could crash. This means that interrupt routines are short and snappy.

* Main IRQ Handling routines, default IRQIV destination *

```
DC93 CLD      ;clear decimal flag
```

```
DC94 LDA &FC  ;get original value of A
DC96 PHA      ;save it
DC97 TXA      ;save X
DC98 PHA      ;
DC99 TYA      ;and Y
DC9A PHA      ;on the stack
              ;note the pre-CMOS code!
DC9B LDA #&DE ;A=&DE
DC9D PHA      ;store it
DC9E LDA #&81 ;save &81
DCA0 PHA      ;store it (a RTS will now jump to DE82)
```

This is quite a useful technique as we will see later. If we now use JMP to go to an OS routine we can ensure that the routine, which ends with an RTS, causes execution to go to a specified point.

This saves a lot of code as it can be arranged that the first device found that called the interrupt will be the only one handled. This, in turn, saves time!

We now poll the hardware looking for who caused it. The first routine deals with the serial/tape system.

```
DCA1 CLV       ;clear V flag
DCA2 LDA &FE08 ;get value of status register of ACIA
DCA5 BVS &DCA9 ;if this was source then DCA9 to process
DCA7 BPL &DD06 ;else if no interrupt requested DD06
DCA9 LDX &EA   ;read RS423 timeout counter
DCAB DEX       ;decrement it
DCAC BMI &DCDE ;and if <0 DCDE
DCAE BVS &DCDD ;else if >&40 DCDD (RTS to DE82)
DCB0 JMP &F588 ;else read ACIA via F588
               ;RTS ends routine!!
DCB3 LDY &FE09 ;read ACIA data
DCB6 ROL       ;
DCB7 ASL       ;
DCB8 TAX       ;X=A
DCB9 TYA       ;A=Y
DCBA LDY #&07  ;Y=07
DCBC JMP &E494 ;check and service EVENT 7 RS423 error
DCBF LDX #&02  ;read RS423 output buffer
DCC1 JSR &E460 ;
DCC4 BCC &DCD6 ;if C=0 buffer is not empty goto DCD6
DCC6 LDA &0285 ;else read printer destination
DCC9 CMP #&02  ;is it serial printer??
DCCB BNE &DC68 ;if not DC68
DCCD INX       ;else X=3
DCCE JSR &E460 ;read printer buffer
DCD1 ROR &02D2 ;rotate to pass carry into bit 7
DCD4 BMI &DC68 ;if set then DC68
DCD6 STA &FE09 ;pass either printer or RS423 data to ACIA
DCD9 LDA #&E7  ;set timeout counter to stored value
DCDB STA &EA   ;
DCDD RTS       ;and exit (to DE82)

               ;A contains ACIA status
DCDE AND &0278 ;AND with ACIA bit mask (normally FF)
DCE1 LSR       ;rotate right to put bit 0 in carry
DCE2 BCC &DCEB ;if carry clear receive register not full so DCEB
DCE4 BVS &DCEB ;if V is set then DCEB
```

```
DCE6 LDY &0250 ;else Y=ACIA control setting
DCE9 BMI &DC7D ;if bit 7 set receive interrupt is enabled so DC7D

DCEB LSR       ;put BIT 2 of ACIA status into
DCEC ROR       ;carry if set then Data Carrier Detected applies
DCED BCS &DCB3 ;jump to DCB3

DCEF BMI &DCBF ;if original bit 1 is set TDR is empty so DCBF
DCF1 BVS &DCDD ;if V is set then exit to DE82

DCF3 LDX #&05  ;X=5
DCF5 JSR &F168 ;issue ROM call 5 'unrecognised ;interrupt'
```

We've seen this ROM service routine call before.

```
DCF8 BEQ &DCDD   ;if a ROM recognises it then exit to DE82
DCFA PLA         ;otherwise get back DE82 address from stack
DCFB PLA         ;
DCFC PLA         ;and get back X, Y and A
DCFD TAY         ;
DCFE PLA         ;
DCFF TAX         ;
DD00 PLA         ;
DD01 STA &FC     ;&FC=A
DD03 JMP (&0206) ;and offer to the user via IRQ2V
```

That was a little convoluted, to say the least. Next week we look at how the
VIAs are dealt with.

BBC 6502 Machine Code
Part Fifteen: Hardware VIA interrupts

After deciding that it wasn't the ACIA that caused the interrupt, the VIAs are the next port of inquisition.

```
* VIA INTERRUPTS ROUTINES *

DD06 LDA &FE4D ;read system VIA interrupt flag register
DD09 BPL &DD47 ;if bit 7=0 the VIA has not caused interrupt goto DD47

DD0B AND &0279 ;mask with VIA bit mask
DD0E AND &FE4E ;and interrupt enable register
DD11 ROR       ;rotate right twice to ;check for IRQ 1 (frame sync)

DD12 ROR       ;
DD13 BCC &DD69 ;if carry clear then no IRQ 1, else IRQ 1 means
               ;interrupt request 1. This is different from the
               ;vector IRQ1.

DD15 DEC &0240 ;decrement vertical sync counter
DD18 LDA &EA   ;A=RS423 Timeout counter
DD1A BPL &DD1E ;if +ve then DD1E
DD1C INC &EA   ;else increment it
DD1E LDA &0251 ;load flash character counter
DD21 BEQ &DD3D ;if 0 then flash system is not in use, ignore it
DD23 DEC &0251 ;else decrement counter
DD26 BNE &DD3D ;and if not 0 go on past reset routine
```

This routine resets the flashing character system.

```
DD28 LDX &0252 ;get mark period count in X
DD2B LDA &0248 ;current VIDEO ULA control setting in A
DD2E LSR       ;shift bit 0 into C to ;check if first colour
DD2F BCC &DD34 ;is effective if so C=0. Jump to DD34
DD31 LDX &0253 ;else get space period count in X
DD34 ROL       ;restore bit
DD35 EOR #&01  ;and invert it
DD37 JSR &EA00 ;then change colour

DD3A STX &0251 ;&0251=X resetting the counter

DD3D LDY #&04  ;Y=4 and call E494 to check and implement vertical
DD3F JSR &E494 ;sync event (4) if necessary
DD42 LDA #&02  ;A=2
DD44 JMP &DE6E ;clear interrupt 1 and exit
```

Remember the RTS routine last time?

```
* PRINTER INTERRUPT USER VIA 1 *

DD47 LDA &FE6D ;Check USER VIA interrupt flags register
DD4A BPL &DCF3 ;if +ve USER VIA did not call interrupt
DD4C AND &0277 ;else check for USER IRQ 1 printer interrupt.
DD4F AND &FE6E ;
DD52 ROR       ;
DD53 ROR       ;
DD54 BCC &DCF3 ;if bit 1=0 then no ;interrupt 1 so DCF3
DD56 LDY &0285 ;else get printer type
DD59 DEY       ;decrement
DD5A BNE &DCF3 ;if not parallel then :CF3
DD5C LDA #&02  ;reset interrupt 1 flag
DD5E STA &FE6D ;
DD61 STA &FE6E ;disable interrupt 1
DD64 LDX #&03  ;and output data to parallel printer
DD66 JMP &E13A ;and exit via RTS

* SYSTEM INTERRUPT 5 Speech *

DD69 ROL       ;get bit 5 into bit 7
DD6A ROL       ;
DD6B ROL       ;
DD6C ROL       ;
DD6D BPL &DDCA ;if not set this is not ;a speech interrupt so DDCA
DD6F LDA #&20  ;
DD71 LDX #&00  ;
DD73 STA &FE4D ;
DD76 STX &FE49 ;and zero high byte of Timer t2
DD79 LDX #&08  ;&FB=8
DD7B STX &FB   ;
DD7D JSR &E45B ;and examine buffer 8
DD80 ROR &02D7 ;shift carry into bit 7
DD83 BMI &DDC9 ;and if set buffer is empty so exit
DD85 TAY       ;else Y=A
DD86 BEQ &DD8D ;
DD88 JSR &EE6D ;control speech chip
DD8B BMI &DDC9 ;if negative exit
DD8D JSR &E460 ;else get a byte from buffer
DD90 STA &F5   ;store it to indicate speech or file ROM
DD92 JSR &E460 ;get another byte
```

```
DD95 STA &F7   ;store it
DD97 JSR &E460 ;and another
DD9A STA &F6   ;giving address to be accessed in paged ROM
DD9C LDY &F5   ;Y=&F5
DD9E BEQ &DDBB ;and if =0 then DDBB
DDA0 BPL &DDB8 ;else if +ve DDB8
DDA2 BIT &F5   ;if bit 6 of F5 =1 (&F5)>&40
DDA4 BVS &DDAB ;then DDAB
DDA6 JSR &EEBB ;else continue for more speech processing
DDA9 BVC &DDB2 ;if bit 6 clear then DDB2
DDAB ASL &F6   ;else double address in &F6/7
DDAD ROL &F7   ;
DDAF JSR &EE3B ;and call EE3B
DDB2 LDY &0261 ;get speech enable/disable flag into Y
DDB5 JMP &EE7F ;and JMP to EE7F

DDB8 JSR &EE7F ;Call EE7F
DDBB LDY &F6   ;get address pointer in Y
DDBD JSR &EE7F ;
DDC0 LDY &F7   ;get address pointer high in Y
DDC2 JSR &EE7F ;
DDC5 LSR &FB   ;
DDC7 BNE &DD7D ;
DDC9 RTS       ;and exit
```

Next week we continue with a look at the remaining System Interrupts.

BBC 6502 Machine Code
Part Sixteen: Timers and Keyboard Interrupts

The last part showed how the BBC Micro handles some of the system interrupt calls. Most of these are pretty routine so we won't continue with an interminable list.

The next interesting routines concern how the timers and keyboard interrupts are handled.

```
* SYSTEM INTERRUPT 6 10mS Clock *

DDCA  BCC  &DE47  ;bit 6 is in carry so if clear there is no 6 so go
                  ;on to DE47
DDCC  LDA  #&40   ;Clear interrupt 6
DDCE  STA  &FE4D  ;
```

This is the start of the update timers routine, This is interesting because of the way that the timer information is stored. It's very clever. There are two timer stores, &292-6 and &297-B. These are updated by adding 1 to the current timer and storing the result in the other, the direction of transfer being changed each time of update.

This ensures that at least one timer is valid at any call as the current timer only is read. Other methods would cause inaccuracies if a timer was read while being updated.

```
DDD1  LDA  &0283   ;get current system clock store pointer (5,or 10)
DDD4  TAX          ;put A in X
DDD5  EOR  #&0F    ;and invert lo nybble (5becomes 10 and vv)
DDD7  PHA          ;store A
DDD8  TAY          ;put A in Y. Carry is always set at this point
DDD9  LDA  &0291,X ;get timer value
```

```
DDDC  ADC   #&00   ;update it
DDDE  STA  &0291,Y ;store result in alternate
DDE1  DEX          ;decrement X
DDE2  BEQ  &DDE7   ;if 0 exit
DDE4  DEY          ;else decrement Y
DDE5  BNE  &DDD9   ;and go back and do next byte
DDE7  PLA          ;get back A
DDE8  STA  &0283   ;and store back in clock pointer (ie. inverse
                   ;previous contents)
DDEB  LDX  #&05    ;set loop pointer for countdown timer
DDED  INC  &029B,X ;increment byte and
DDF0  BNE  &DDFA   ;if not 0 then DDFA
DDF2  DEX          ;else decrement pointer
DDF3  BNE  &DDED   ;and if not 0 do it again
DDF5  LDY  #&05    ;process EVENT 5 interrupt timer
DDF7  JSR  &E494   ;
DDFA  LDA  &02B1   ;get byte of inkey countdown timer
DDFD  BNE  &DE07   ;if not 0 then DE07
DDFF  LDA  &02B2   ;else get next byte
DE02  BEQ  &DE0A   ;if 0 DE0A
DE04  DEC  &02B2   ;decrement 2B2
DE07  DEC  &02B1   ;and 2B1
DE0A  BIT  &02CE   ;read bit 7 of envelope processing byte
DE0D  BPL  &DE1A   ;if 0 then DE1A
DE0F  INC  &02CE   ;else increment to 0
DE12  CLI          ;allow interrupts
DE13  JSR  &EB47   ;and do routine sound processes
DE16  SEI          ;bar interrupts
DE17  DEC  &02CE   ;DEC envelope processing byte back to 0
DE1A  BIT  &02D7   ;read speech buffer busy flag
DE1D  BMI  &DE2B   ;if set speech buffer is empty, skip routine
DE1F  JSR  &EE6D   ;update speech system variables
DE22  EOR  #&A0    ;
DE24  CMP  #&60    ;
DE26  BCC  &DE2B   ;if result >=&60 DE2B
DE28  JSR  &DD79   ;else more speech work
DE2B  BIT  &D9B7   ;set V and C
DE2E  JSR  &DCA2   ;check if ACIA needs attention
DE31  LDA  &EC     ;check if key has been pressed
DE33  ORA  &ED     ;
DE35  AND  &0242   ;(this is 0 if keyboard is to be ignored, else
                   ;&FF)
DE38  BEQ  &DE3E   ;if 0 ignore keyboard
DE3A  SEC          ;else set carry
DE3B  JSR  &F065   ;and call keyboard
DE3E  JSR  &E19B   ;check for data in use defined printer channel
DE41  BIT  &FEC0   ;if ADC bit 6 is set ADC is not busy
DE44  BVS  &DE4A   ;so DE4A
DE46  RTS          ;else return

* SYSTEM INTERRUPT 4 ADC end of conversion *

DE47  ROL          ;put original bit 4 from FE4D into bit 7 of A
DE48  BPL  &DE72   ;if not set DE72
DE4A  LDX  &024C   ;else get current ADC channel
DE4D  BEQ  &DE6C   ;if 0 DE6C
DE4F  LDA  &FEC2   ;read low data byte
DE52  STA  &02B5,X ;store it in &2B6,7,8 or 9
DE55  LDA  &FEC1   ;get high data byte
DE58  STA  &02B9,X ;and store it in hi byte
```

```
DE5B  STX  &02BE   ;store in Analogue system flag marking last channel
DE5E  LDY  #&03    ;handle event 3 conversion complete
DE60  JSR  &E494   ;
DE63  DEX          ;decrement X
DE64  BNE  &DE69   ;if X=0
DE66  LDX  &024D   ;get highest ADC channel present
DE69  JSR  &DE8F   ;and start new conversion
DE6C  LDA  #&10    ;reset interrupt 4
DE6E  STA  &FE4D   ;
DE71  RTS          ;and return

* SYSTEM INTERRUPT 0 Keyboard *

DE72  ROL         ;get original bit 0 in bit 7 position
DE73  ROL         ;
DE74  ROL         ;
DE75  ROL         ;
DE76  BPL  &DE7F  ;if bit 7 clear not a keyboard interrupt
DE78  JSR  &F065  ;else scan keyboard
DE7B  LDA  #&01   ;A=1
DE7D  BNE  &DE6E  ;and off to reset interrupt and exit
DE7F  JMP  &DCF3  ;and again a subroutine to exit.
```

Now we come to the point you've all been waiting for. This mystery RTSreturns all subroutines to &DE82.

```
************** exit routine
DE82  PLA         ;restore registers
DE83  TAY         ;
DE84  PLA         ;
DE85  TAX         ;
DE86  PLA         ;
DE87  STA   &FC   ;store A

* IRQ2V default entry *

DE89  LDA   &FC   ;get back original value of A
DE8B  RTI         ;and return to calling routine.
```

NEXT WEEK: OSBYTE entry.

BBC 6502 Machine Code
Part Seventeen: The BBC Operating System

We've been examining the BBC operating system in some detail over the last few weeks. Unfortunately the demise of Micronet means that we cannot finish completely, as we hoped. So we've put together the next twenty weeks' articles in the form of a completely commented disassembly of OS 1.20.

This is an excellent example of BBC programming and is full of tips.

Just to remind you of the main points of the software. Entry points are pointed to by a jump table in the last six bytes of the ROM.

The font characters are located from &C000 to &C2FF.

OK, so here it is all commented and ready for you to peruse.

Ed says: I have uploaded the series of disassembly articles as ten

short TSW files. Look on Micronet on 700100239 (before it's too late!)

********** THE END **********

```
 ***************** VDU CHARACTER FONT LOOK UP TABLE
****************************

 C000  DB  00  ;00000000   ........
 C001  DB  00  ;00000000   ........
 C002  DB  00  ;00000000   ........
 C003  DB  00  ;00000000   ........
 C004  DB  00  ;00000000   ........
 C005  DB  00  ;00000000   ........
 C006  DB  00  ;00000000   ........
 C007  DB  00  ;00000000   ........

 C008  DB  18  ;00011000   ...**...
 C009  DB  18  ;00011000   ...**...
 C00A  DB  18  ;00011000   ...**...
 C00B  DB  18  ;00011000   ...**...
 C00C  DB  18  ;00011000   ...**...
 C00D  DB  00  ;00000000   ........
 C00E  DB  18  ;00011000   ...**...
 C00F  DB  00  ;00000000   ........

 C010  DB  6C  ;01101100   .**.**..
 C011  DB  6C  ;01101100   .**.**..
 C012  DB  6C  ;01101100   .**.**..
 C013  DB  00  ;00000000   ........
 C014  DB  00  ;00000000   ........
 C015  DB  00  ;00000000   ........
 C016  DB  00  ;00000000   ........
 C017  DB  00  ;00000000   ........

 C018  DB  36  ;00110110   ..**.**.
 C019  DB  36  ;00110110   ..**.**.
 C01A  DB  7F  ;01111111   .*******
 C01B  DB  36  ;00110110   ..**.**.
 C01C  DB  7F  ;01111111   .*******
 C01D  DB  36  ;00110110   ..**.**.
 C01E  DB  36  ;00110110   ..**.**.
 C01F  DB  00  ;00000000   ........

 C020  DB  0C  ;00001100   ....**..
 C021  DB  3F  ;00111111   ..******
 C022  DB  68  ;01101000   .**.*...
 C023  DB  3E  ;00111110   ..*****.
 C024  DB  0B  ;00001011   ....*.**
 C025  DB  7E  ;01111110   .******.
 C026  DB  18  ;00011000   ...**...
 C027  DB  00  ;00000000   ........

 C028  DB  60  ;01100000   .**.....
 C029  DB  66  ;01100110   .**..**.
 C02A  DB  0C  ;00001100   ....**..
 C02B  DB  18  ;00011000   ...**...
```

```
C02C  DB  30  ;00110000   ..**....
C02D  DB  66  ;01100110   .**..**.
C02E  DB  06  ;00000110   .....**.
C02F  DB  00  ;00000000   ........

C030  DB  38  ;00111000   ..***...
C031  DB  6C  ;01101100   .**.**..
C032  DB  6C  ;01101100   .**.**..
C033  DB  38  ;00111000   ..***...
C034  DB  6D  ;01101101   .**.**.*
C035  DB  66  ;01100110   .**..**.
C036  DB  3B  ;00111011   ..***.**
C037  DB  00  ;00000000   ........

C038  DB  0C  ;00001100   ....**..
C039  DB  18  ;00011000   ...**...
C03A  DB  30  ;00110000   ..**....
C03B  DB  00  ;00000000   ........
C03C  DB  00  ;00000000   ........
C03D  DB  00  ;00000000   ........
C03E  DB  00  ;00000000   ........
C03F  DB  00  ;00000000   ........

C040  DB  0C  ;00001100   ....**..
C041  DB  18  ;00011000   ...**...
C042  DB  30  ;00110000   ..**....
C043  DB  30  ;00110000   ..**....
C044  DB  30  ;00110000   ..**....
C045  DB  18  ;00011000   ...**...
C046  DB  0C  ;00001100   ....**..
C047  DB  00  ;00000000   ........

C048  DB  30  ;00110000   ..**....
C049  DB  18  ;00011000   ...**...
C04A  DB  0C  ;00001100   ....**..
C04B  DB  0C  ;00001100   ....**..
C04C  DB  0C  ;00001100   ....**..
C04D  DB  18  ;00011000   ...**...
C04E  DB  30  ;00110000   ..**....
C04F  DB  00  ;00000000   ........

C050  DB  00  ;00000000   ........
C051  DB  18  ;00011000   ...**...
C052  DB  7E  ;01111110   .******.
C053  DB  3C  ;00111100   ..****..
C054  DB  7E  ;01111110   .******.
C055  DB  18  ;00011000   ...**...
C056  DB  00  ;00000000   ........
C057  DB  00  ;00000000   ........

C058  DB  00  ;00000000   ........
C059  DB  18  ;00011000   ...**...
C05A  DB  18  ;00011000   ...**...
C05B  DB  7E  ;01111110   .******.
```

```
C05C  DB  18  ;00011000   ...**...
C05D  DB  18  ;00011000   ...**...
C05E  DB  00  ;00000000   ........
C05F  DB  00  ;00000000   ........

C060  DB  00  ;00000000   ........
C061  DB  00  ;00000000   ........
C062  DB  00  ;00000000   ........
C063  DB  00  ;00000000   ........
C064  DB  00  ;00000000   ........
C065  DB  18  ;00011000   ...**...
C066  DB  18  ;00011000   ...**...
C067  DB  30  ;00110000   ..**....

C068  DB  00  ;00000000   ........
C069  DB  00  ;00000000   ........
C06A  DB  00  ;00000000   ........
C06B  DB  7E  ;01111110   .******.
C06C  DB  00  ;00000000   ........
C06D  DB  00  ;00000000   ........
C06E  DB  00  ;00000000   ........
C06F  DB  00  ;00000000   ........

C070  DB  00  ;00000000   ........
C071  DB  00  ;00000000   ........
C072  DB  00  ;00000000   ........
C073  DB  00  ;00000000   ........
C074  DB  00  ;00000000   ........
C075  DB  18  ;00011000   ...**...
C076  DB  18  ;00011000   ...**...
C077  DB  00  ;00000000   ........

C078  DB  00  ;00000000   ........
C079  DB  06  ;00000110   .....**.
C07A  DB  0C  ;00001100   ....**..
C07B  DB  18  ;00011000   ...**...
C07C  DB  30  ;00110000   ..**....
C07D  DB  60  ;01100000   .**.....
C07E  DB  00  ;00000000   ........
C07F  DB  00  ;00000000   ........

C080  DB  3C  ;00111100   ..****..
C081  DB  66  ;01100110   .**..**.
C082  DB  6E  ;01101110   .**.***.
C083  DB  7E  ;01111110   .******.
C084  DB  76  ;01110110   .***.**.
C085  DB  66  ;01100110   .**..**.
C086  DB  3C  ;00111100   ..****..
C087  DB  00  ;00000000   ........

C088  DB  18  ;00011000   ...**...
C089  DB  38  ;00111000   ..***...
C08A  DB  18  ;00011000   ...**...
C08B  DB  18  ;00011000   ...**...
```

```
C08C  DB  18  ;00011000   ...**...
C08D  DB  18  ;00011000   ...**...
C08E  DB  7E  ;01111110   .******.
C08F  DB  00  ;00000000   ........

C090  DB  3C  ;00111100   ..****..
C091  DB  66  ;01100110   .**..**.
C092  DB  06  ;00000110   .....**.
C093  DB  0C  ;00001100   ....**..
C094  DB  18  ;00011000   ...**...
C095  DB  30  ;00110000   ..**....
C096  DB  7E  ;01111110   .******.
C097  DB  00  ;00000000   ........

C098  DB  3C  ;00111100   ..****..
C099  DB  66  ;01100110   .**..**.
C09A  DB  06  ;00000110   .....**.
C09B  DB  1C  ;00011100   ...***..
C09C  DB  06  ;00000110   .....**.
C09D  DB  66  ;01100110   .**..**.
C09E  DB  3C  ;00111100   ..****..
C09F  DB  00  ;00000000   ........

C0A0  DB  0C  ;00001100   ....**..
C0A1  DB  1C  ;00011100   ...***..
C0A2  DB  3C  ;00111100   ..****..
C0A3  DB  6C  ;01101100   .**.**..
C0A4  DB  7E  ;01111110   .******.
C0A5  DB  0C  ;00001100   ....**..
C0A6  DB  0C  ;00001100   ....**..
C0A7  DB  00  ;00000000   ........

C0A8  DB  7E  ;01111110   .******.
C0A9  DB  60  ;01100000   .**.....
C0AA  DB  7C  ;01111100   .*****..
C0AB  DB  06  ;00000110   .....**.
C0AC  DB  06  ;00000110   .....**.
C0AD  DB  66  ;01100110   .**..**.
C0AE  DB  3C  ;00111100   ..****..
C0AF  DB  00  ;00000000   ........

C0B0  DB  1C  ;00011100   ...***..
C0B1  DB  30  ;00110000   ..**....
C0B2  DB  60  ;01100000   .**.....
C0B3  DB  7C  ;01111100   .*****..
C0B4  DB  66  ;01100110   .**..**.
C0B5  DB  66  ;01100110   .**..**.
C0B6  DB  3C  ;00111100   ..****..
C0B7  DB  00  ;00000000   ........

C0B8  DB  7E  ;01111110   .******.
C0B9  DB  06  ;00000110   .....**.
C0BA  DB  0C  ;00001100   ....**..
C0BB  DB  18  ;00011000   ...**...
```

```
C0BC  DB  30  ;00110000   ..**....
C0BD  DB  30  ;00110000   ..**....
C0BE  DB  30  ;00110000   ..**....
C0BF  DB  00  ;00000000   ........

C0C0  DB  3C  ;00111100   ..****..
C0C1  DB  66  ;01100110   .**..**.
C0C2  DB  66  ;01100110   .**..**.
C0C3  DB  3C  ;00111100   ..****..
C0C4  DB  66  ;01100110   .**..**.
C0C5  DB  66  ;01100110   .**..**.
C0C6  DB  3C  ;00111100   ..****..
C0C7  DB  00  ;00000000   ........

C0C8  DB  3C  ;00111100   ..****..
C0C9  DB  66  ;01100110   .**..**.
C0CA  DB  66  ;01100110   .**..**.
C0CB  DB  3E  ;00111110   ..*****.
C0CC  DB  06  ;00000110   .....**.
C0CD  DB  0C  ;00001100   ....**..
C0CE  DB  38  ;00111000   ..***...
C0CF  DB  00  ;00000000   ........

C0D0  DB  00  ;00000000   ........
C0D1  DB  00  ;00000000   ........
C0D2  DB  18  ;00011000   ...**...
C0D3  DB  18  ;00011000   ...**...
C0D4  DB  00  ;00000000   ........
C0D5  DB  18  ;00011000   ...**...
C0D6  DB  18  ;00011000   ...**...
C0D7  DB  00  ;00000000   ........

C0D8  DB  00  ;00000000   ........
C0D9  DB  00  ;00000000   ........
C0DA  DB  18  ;00011000   ...**...
C0DB  DB  18  ;00011000   ...**...
C0DC  DB  00  ;00000000   ........
C0DD  DB  18  ;00011000   ...**...
C0DE  DB  18  ;00011000   ...**...
C0DF  DB  30  ;00110000   ..**....

C0E0  DB  0C  ;00001100   ....**..
C0E1  DB  18  ;00011000   ...**...
C0E2  DB  30  ;00110000   ..**....
C0E3  DB  60  ;01100000   .**.....
C0E4  DB  30  ;00110000   ..**....
C0E5  DB  18  ;00011000   ...**...
C0E6  DB  0C  ;00001100   ....**..
C0E7  DB  00  ;00000000   ........

C0E8  DB  00  ;00000000   ........
C0E9  DB  00  ;00000000   ........
C0EA  DB  7E  ;01111110   .******.
C0EB  DB  00  ;00000000   ........
```

```
C0EC  DB  7E  ;01111110   .******.
C0ED  DB  00  ;00000000   ........
C0EE  DB  00  ;00000000   ........
C0EF  DB  00  ;00000000   ........

C0F0  DB  30  ;00110000   ..**....
C0F1  DB  18  ;00011000   ...**...
C0F2  DB  0C  ;00001100   ....**..
C0F3  DB  06  ;00000110   .....**.
C0F4  DB  0C  ;00001100   ....**..
C0F5  DB  18  ;00011000   ...**...
C0F6  DB  30  ;00110000   ..**....
C0F7  DB  00  ;00000000   ........

C0F8  DB  3C  ;00111100   ..****..
C0F9  DB  66  ;01100110   .**..**.
C0FA  DB  0C  ;00001100   ....**..
C0FB  DB  18  ;00011000   ...**...
C0FC  DB  18  ;00011000   ...**...
C0FD  DB  00  ;00000000   ........
C0FE  DB  18  ;00011000   ...**...
C0FF  DB  00  ;00000000   ........

C100  DB  3C  ;00111100   ..****..
C101  DB  66  ;01100110   .**..**.
C102  DB  6E  ;01101110   .**.***.
C103  DB  6A  ;01101010   .**.*.*.
C104  DB  6E  ;01101110   .**.***.
C105  DB  60  ;01100000   .**.....
C106  DB  3C  ;00111100   ..****..
C107  DB  00  ;00000000   ........

C108  DB  3C  ;00111100   ..****..
C109  DB  66  ;01100110   .**..**.
C10A  DB  66  ;01100110   .**..**.
C10B  DB  7E  ;01111110   .******.
C10C  DB  66  ;01100110   .**..**.
C10D  DB  66  ;01100110   .**..**.
C10E  DB  66  ;01100110   .**..**.
C10F  DB  00  ;00000000   ........

C110  DB  7C  ;01111100   .*****..
C111  DB  66  ;01100110   .**..**.
C112  DB  66  ;01100110   .**..**.
C113  DB  7C  ;01111100   .*****..
C114  DB  66  ;01100110   .**..**.
C115  DB  66  ;01100110   .**..**.
C116  DB  7C  ;01111100   .*****..
C117  DB  00  ;00000000   ........

C118  DB  3C  ;00111100   ..****..
C119  DB  66  ;01100110   .**..**.
C11A  DB  60  ;01100000   .**.....
C11B  DB  60  ;01100000   .**.....
```

```
C11C  DB  60  ;01100000   .**.....
C11D  DB  66  ;01100110   .**..**.
C11E  DB  3C  ;00111100   ..****..
C11F  DB  00  ;00000000   ........

C120  DB  78  ;01111000   .****...
C121  DB  6C  ;01101100   .**.**..
C122  DB  66  ;01100110   .**..**.
C123  DB  66  ;01100110   .**..**.
C124  DB  66  ;01100110   .**..**.
C125  DB  6C  ;01101100   .**.**..
C126  DB  78  ;01111000   .****...
C127  DB  00  ;00000000   ........

C128  DB  7E  ;01111110   .******.
C129  DB  60  ;01100000   .**.....
C12A  DB  60  ;01100000   .**.....
C12B  DB  7C  ;01111100   .*****..
C12C  DB  60  ;01100000   .**.....
C12D  DB  60  ;01100000   .**.....
C12E  DB  7E  ;01111110   .******.
C12F  DB  00  ;00000000   ........

C130  DB  7E  ;01111110   .******.
C131  DB  60  ;01100000   .**.....
C132  DB  60  ;01100000   .**.....
C133  DB  7C  ;01111100   .*****..
C134  DB  60  ;01100000   .**.....
C135  DB  60  ;01100000   .**.....
C136  DB  60  ;01100000   .**.....
C137  DB  00  ;00000000   ........

C138  DB  3C  ;00111100   ..****..
C139  DB  66  ;01100110   .**..**.
C13A  DB  60  ;01100000   .**.....
C13B  DB  6E  ;01101110   .**.***.
C13C  DB  66  ;01100110   .**..**.
C13D  DB  66  ;01100110   .**..**.
C13E  DB  3C  ;00111100   ..****..
C13F  DB  00  ;00000000   ........

C140  DB  66  ;01100110   .**..**.
C141  DB  66  ;01100110   .**..**.
C142  DB  66  ;01100110   .**..**.
C143  DB  7E  ;01111110   .******.
C144  DB  66  ;01100110   .**..**.
C145  DB  66  ;01100110   .**..**.
C146  DB  66  ;01100110   .**..**.
C147  DB  00  ;00000000   ........

C148  DB  7E  ;01111110   .******.
C149  DB  18  ;00011000   ...**...
C14A  DB  18  ;00011000   ...**...
C14B  DB  18  ;00011000   ...**...
```

```
C14C  DB  18  ;00011000   ...**...
C14D  DB  18  ;00011000   ...**...
C14E  DB  7E  ;01111110   .******.
C14F  DB  00  ;00000000   ........

C150  DB  3E  ;00111110   ..*****.
C151  DB  0C  ;00001100   ....**..
C152  DB  0C  ;00001100   ....**..
C153  DB  0C  ;00001100   ....**..
C154  DB  0C  ;00001100   ....**..
C155  DB  6C  ;01101100   .**.**..
C156  DB  38  ;00111000   ..***...
C157  DB  00  ;00000000   ........

C158  DB  66  ;01100110   .**..**.
C159  DB  6C  ;01101100   .**.**..
C15A  DB  78  ;01111000   .****...
C15B  DB  70  ;01110000   .***....
C15C  DB  78  ;01111000   .****...
C15D  DB  6C  ;01101100   .**.**..
C15E  DB  66  ;01100110   .**..**.
C15F  DB  00  ;00000000   ........

C160  DB  60  ;01100000   .**.....
C161  DB  60  ;01100000   .**.....
C162  DB  60  ;01100000   .**.....
C163  DB  60  ;01100000   .**.....
C164  DB  60  ;01100000   .**.....
C165  DB  60  ;01100000   .**.....
C166  DB  7E  ;01111110   .******.
C167  DB  00  ;00000000   ........

C168  DB  63  ;01100011   .**...**
C169  DB  77  ;01110111   .***.***
C16A  DB  7F  ;01111111   .*******
C16B  DB  6B  ;01101011   .**.*.**
C16C  DB  6B  ;01101011   .**.*.**
C16D  DB  63  ;01100011   .**...**
C16E  DB  63  ;01100011   .**...**
C16F  DB  00  ;00000000   ........

C170  DB  66  ;01100110   .**..**.
C171  DB  66  ;01100110   .**..**.
C172  DB  76  ;01110110   .***.**.
C173  DB  7E  ;01111110   .******.
C174  DB  6E  ;01101110   .**.***.
C175  DB  66  ;01100110   .**..**.
C176  DB  66  ;01100110   .**..**.
C177  DB  00  ;00000000   ........

C178  DB  3C  ;00111100   ..****..
C179  DB  66  ;01100110   .**..**.
C17A  DB  66  ;01100110   .**..**.
C17B  DB  66  ;01100110   .**..**.
```

```
C17C  DB  66  ;01100110   .**..**.
C17D  DB  66  ;01100110   .**..**.
C17E  DB  3C  ;00111100   ..****..
C17F  DB  00  ;00000000   ........

C180  DB  7C  ;01111100   .*****..
C181  DB  66  ;01100110   .**..**.
C182  DB  66  ;01100110   .**..**.
C183  DB  7C  ;01111100   .*****..
C184  DB  60  ;01100000   .**.....
C185  DB  60  ;01100000   .**.....
C186  DB  60  ;01100000   .**.....
C187  DB  00  ;00000000   ........

C188  DB  3C  ;00111100   ..****..
C189  DB  66  ;01100110   .**..**.
C18A  DB  66  ;01100110   .**..**.
C18B  DB  66  ;01100110   .**..**.
C18C  DB  6A  ;01101010   .**.*.*.
C18D  DB  6C  ;01101100   .**.**..
C18E  DB  36  ;00110110   ..**.**.
C18F  DB  00  ;00000000   ........

C190  DB  7C  ;01111100   .*****..
C191  DB  66  ;01100110   .**..**.
C192  DB  66  ;01100110   .**..**.
C193  DB  7C  ;01111100   .*****..
C194  DB  6C  ;01101100   .**.**..
C195  DB  66  ;01100110   .**..**.
C196  DB  66  ;01100110   .**..**.
C197  DB  00  ;00000000   ........

C198  DB  3C  ;00111100   ..****..
C199  DB  66  ;01100110   .**..**.
C19A  DB  60  ;01100000   .**.....
C19B  DB  3C  ;00111100   ..****..
C19C  DB  06  ;00000110   .....**.
C19D  DB  66  ;01100110   .**..**.
C19E  DB  3C  ;00111100   ..****..
C19F  DB  00  ;00000000   ........

C1A0  DB  7E  ;01111110   .******.
C1A1  DB  18  ;00011000   ...**...
C1A2  DB  18  ;00011000   ...**...
C1A3  DB  18  ;00011000   ...**...
C1A4  DB  18  ;00011000   ...**...
C1A5  DB  18  ;00011000   ...**...
C1A6  DB  18  ;00011000   ...**...
C1A7  DB  00  ;00000000   ........

C1A8  DB  66  ;01100110   .**..**.
C1A9  DB  66  ;01100110   .**..**.
C1AA  DB  66  ;01100110   .**..**.
C1AB  DB  66  ;01100110   .**..**.
```

```
C1AC  DB  66  ;01100110   .**..**.
C1AD  DB  66  ;01100110   .**..**.
C1AE  DB  3C  ;00111100   ..****..
C1AF  DB  00  ;00000000   ........

C1B0  DB  66  ;01100110   .**..**.
C1B1  DB  66  ;01100110   .**..**.
C1B2  DB  66  ;01100110   .**..**.
C1B3  DB  66  ;01100110   .**..**.
C1B4  DB  66  ;01100110   .**..**.
C1B5  DB  3C  ;00111100   ..****..
C1B6  DB  18  ;00011000   ...**...
C1B7  DB  00  ;00000000   ........

C1B8  DB  63  ;01100011   .**...**
C1B9  DB  63  ;01100011   .**...**
C1BA  DB  6B  ;01101011   .**.*.**
C1BB  DB  6B  ;01101011   .**.*.**
C1BC  DB  7F  ;01111111   .*******
C1BD  DB  77  ;01110111   .***.***
C1BE  DB  63  ;01100011   .**...**
C1BF  DB  00  ;00000000   ........

C1C0  DB  66  ;01100110   .**..**.
C1C1  DB  66  ;01100110   .**..**.
C1C2  DB  3C  ;00111100   ..****..
C1C3  DB  18  ;00011000   ...**...
C1C4  DB  3C  ;00111100   ..****..
C1C5  DB  66  ;01100110   .**..**.
C1C6  DB  66  ;01100110   .**..**.
C1C7  DB  00  ;00000000   ........

C1C8  DB  66  ;01100110   .**..**.
C1C9  DB  66  ;01100110   .**..**.
C1CA  DB  66  ;01100110   .**..**.
C1CB  DB  3C  ;00111100   ..****..
C1CC  DB  18  ;00011000   ...**...
C1CD  DB  18  ;00011000   ...**...
C1CE  DB  18  ;00011000   ...**...
C1CF  DB  00  ;00000000   ........

C1D0  DB  7E  ;01111110   .******.
C1D1  DB  06  ;00000110   .....**.
C1D2  DB  0C  ;00001100   ....**..
C1D3  DB  18  ;00011000   ...**...
C1D4  DB  30  ;00110000   ..**....
C1D5  DB  60  ;01100000   .**.....
C1D6  DB  7E  ;01111110   .******.
C1D7  DB  00  ;00000000   ........

C1D8  DB  7C  ;01111100   .*****..
C1D9  DB  60  ;01100000   .**.....
C1DA  DB  60  ;01100000   .**.....
C1DB  DB  60  ;01100000   .**.....
```

```
C1DC  DB  60  ;01100000   .**.....
C1DD  DB  60  ;01100000   .**.....
C1DE  DB  7C  ;01111100   .*****..
C1DF  DB  00  ;00000000   ........

C1E0  DB  00  ;00000000   ........
C1E1  DB  60  ;01100000   .**.....
C1E2  DB  30  ;00110000   ..**....
C1E3  DB  18  ;00011000   ...**...
C1E4  DB  0C  ;00001100   ....**..
C1E5  DB  06  ;00000110   .....**.
C1E6  DB  00  ;00000000   ........
C1E7  DB  00  ;00000000   ........

C1E8  DB  3E  ;00111110   ..*****.
C1E9  DB  06  ;00000110   .....**.
C1EA  DB  06  ;00000110   .....**.
C1EB  DB  06  ;00000110   .....**.
C1EC  DB  06  ;00000110   .....**.
C1ED  DB  06  ;00000110   .....**.
C1EE  DB  3E  ;00111110   ..*****.
C1EF  DB  00  ;00000000   ........

C1F0  DB  18  ;00011000   ...**...
C1F1  DB  3C  ;00111100   ..****..
C1F2  DB  66  ;01100110   .**..**.
C1F3  DB  42  ;01000010   .*....*.
C1F4  DB  00  ;00000000   ........
C1F5  DB  00  ;00000000   ........
C1F6  DB  00  ;00000000   ........
C1F7  DB  00  ;00000000   ........

C1F8  DB  00  ;00000000   ........
C1F9  DB  00  ;00000000   ........
C1FA  DB  00  ;00000000   ........
C1FB  DB  00  ;00000000   ........
C1FC  DB  00  ;00000000   ........
C1FD  DB  00  ;00000000   ........
C1FE  DB  00  ;00000000   ........
C1FF  DB  FF  ;11111111   ********

C200  DB  1C  ;00011100   ...***..
C201  DB  36  ;00110110   ..**.**.
C202  DB  30  ;00110000   ..**....
C203  DB  7C  ;01111100   .*****..
C204  DB  30  ;00110000   ..**....
C205  DB  30  ;00110000   ..**....
C206  DB  7E  ;01111110   .******.
C207  DB  00  ;00000000   ........

C208  DB  00  ;00000000   ........
C209  DB  00  ;00000000   ........
C20A  DB  3C  ;00111100   ..****..
C20B  DB  06  ;00000110   .....**.
```

```
C20C  DB  3E  ;00111110   ..*****.
C20D  DB  66  ;01100110   .**..**.
C20E  DB  3E  ;00111110   ..*****.
C20F  DB  00  ;00000000   ........

C210  DB  60  ;01100000   .**.....
C211  DB  60  ;01100000   .**.....
C212  DB  7C  ;01111100   .*****..
C213  DB  66  ;01100110   .**..**.
C214  DB  66  ;01100110   .**..**.
C215  DB  66  ;01100110   .**..**.
C216  DB  7C  ;01111100   .*****..
C217  DB  00  ;00000000   ........

C218  DB  00  ;00000000   ........
C219  DB  00  ;00000000   ........
C21A  DB  3C  ;00111100   ..****..
C21B  DB  66  ;01100110   .**..**.
C21C  DB  60  ;01100000   .**.....
C21D  DB  66  ;01100110   .**..**.
C21E  DB  3C  ;00111100   ..****..
C21F  DB  00  ;00000000   ........

C220  DB  06  ;00000110   .....**.
C221  DB  06  ;00000110   .....**.
C222  DB  3E  ;00111110   ..*****.
C223  DB  66  ;01100110   .**..**.
C224  DB  66  ;01100110   .**..**.
C225  DB  66  ;01100110   .**..**.
C226  DB  3E  ;00111110   ..*****.
C227  DB  00  ;00000000   ........

C228  DB  00  ;00000000   ........
C229  DB  00  ;00000000   ........
C22A  DB  3C  ;00111100   ..****..
C22B  DB  66  ;01100110   .**..**.
C22C  DB  7E  ;01111110   .******.
C22D  DB  60  ;01100000   .**.....
C22E  DB  3C  ;00111100   ..****..
C22F  DB  00  ;00000000   ........

C230  DB  1C  ;00011100   ...***..
C231  DB  30  ;00110000   ..**....
C232  DB  30  ;00110000   ..**....
C233  DB  7C  ;01111100   .*****..
C234  DB  30  ;00110000   ..**....
C235  DB  30  ;00110000   ..**....
C236  DB  30  ;00110000   ..**....
C237  DB  00  ;00000000   ........

C238  DB  00  ;00000000   ........
C239  DB  00  ;00000000   ........
C23A  DB  3E  ;00111110   ..*****.
C23B  DB  66  ;01100110   .**..**.
```

```
C23C  DB  66  ;01100110   .**..**.
C23D  DB  3E  ;00111110   ..*****.
C23E  DB  06  ;00000110   .....**.
C23F  DB  3C  ;00111100   ..****..

C240  DB  60  ;01100000   .**.....
C241  DB  60  ;01100000   .**.....
C242  DB  7C  ;01111100   .*****..
C243  DB  66  ;01100110   .**..**.
C244  DB  66  ;01100110   .**..**.
C245  DB  66  ;01100110   .**..**.
C246  DB  66  ;01100110   .**..**.
C247  DB  00  ;00000000   ........

C248  DB  18  ;00011000   ...**...
C249  DB  00  ;00000000   ........
C24A  DB  38  ;00111000   ..***...
C24B  DB  18  ;00011000   ...**...
C24C  DB  18  ;00011000   ...**...
C24D  DB  18  ;00011000   ...**...
C24E  DB  3C  ;00111100   ..****..
C24F  DB  00  ;00000000   ........

C250  DB  18  ;00011000   ...**...
C251  DB  00  ;00000000   ........
C252  DB  38  ;00111000   ..***...
C253  DB  18  ;00011000   ...**...
C254  DB  18  ;00011000   ...**...
C255  DB  18  ;00011000   ...**...
C256  DB  18  ;00011000   ...**...
C257  DB  70  ;01110000   .***....

C258  DB  60  ;01100000   .**.....
C259  DB  60  ;01100000   .**.....
C25A  DB  66  ;01100110   .**..**.
C25B  DB  6C  ;01101100   .**.**..
C25C  DB  78  ;01111000   .****...
C25D  DB  6C  ;01101100   .**.**..
C25E  DB  66  ;01100110   .**..**.
C25F  DB  00  ;00000000   ........

C260  DB  38  ;00111000   ..***...
C261  DB  18  ;00011000   ...**...
C262  DB  18  ;00011000   ...**...
C263  DB  18  ;00011000   ...**...
C264  DB  18  ;00011000   ...**...
C265  DB  18  ;00011000   ...**...
C266  DB  3C  ;00111100   ..****..
C267  DB  00  ;00000000   ........

C268  DB  00  ;00000000   ........
C269  DB  00  ;00000000   ........
C26A  DB  36  ;00110110   ..**.**.
C26B  DB  7F  ;01111111   .*******
```

```
C26C  DB  6B  ;01101011   .**.*.**
C26D  DB  6B  ;01101011   .**.*.**
C26E  DB  63  ;01100011   .**...**
C26F  DB  00  ;00000000   ........

C270  DB  00  ;00000000   ........
C271  DB  00  ;00000000   ........
C272  DB  7C  ;01111100   .*****..
C273  DB  66  ;01100110   .**..**.
C274  DB  66  ;01100110   .**..**.
C275  DB  66  ;01100110   .**..**.
C276  DB  66  ;01100110   .**..**.
C277  DB  00  ;00000000   ........

C278  DB  00  ;00000000   ........
C279  DB  00  ;00000000   ........
C27A  DB  3C  ;00111100   ..****..
C27B  DB  66  ;01100110   .**..**.
C27C  DB  66  ;01100110   .**..**.
C27D  DB  66  ;01100110   .**..**.
C27E  DB  3C  ;00111100   ..****..
C27F  DB  00  ;00000000   ........

C280  DB  00  ;00000000   ........
C281  DB  00  ;00000000   ........
C282  DB  7C  ;01111100   .*****..
C283  DB  66  ;01100110   .**..**.
C284  DB  66  ;01100110   .**..**.
C285  DB  7C  ;01111100   .*****..
C286  DB  60  ;01100000   .**.....
C287  DB  60  ;01100000   .**.....

C288  DB  00  ;00000000   ........
C289  DB  00  ;00000000   ........
C28A  DB  3E  ;00111110   ..*****.
C28B  DB  66  ;01100110   .**..**.
C28C  DB  66  ;01100110   .**..**.
C28D  DB  3E  ;00111110   ..*****.
C28E  DB  06  ;00000110   .....**.
C28F  DB  07  ;00000111   .....***

C290  DB  00  ;00000000   ........
C291  DB  00  ;00000000   ........
C292  DB  6C  ;01101100   .**.**..
C293  DB  76  ;01110110   .***.**.
C294  DB  60  ;01100000   .**.....
C295  DB  60  ;01100000   .**.....
C296  DB  60  ;01100000   .**.....
C297  DB  00  ;00000000   ........

C298  DB  00  ;00000000   ........
C299  DB  00  ;00000000   ........
C29A  DB  3E  ;00111110   ..*****.
C29B  DB  60  ;01100000   .**.....
```

```
C29C  DB  3C  ;00111100   ..****..
C29D  DB  06  ;00000110   .....**.
C29E  DB  7C  ;01111100   .*****..
C29F  DB  00  ;00000000   ........

C2A0  DB  30  ;00110000   ..**....
C2A1  DB  30  ;00110000   ..**....
C2A2  DB  7C  ;01111100   .*****..
C2A3  DB  30  ;00110000   ..**....
C2A4  DB  30  ;00110000   ..**....
C2A5  DB  30  ;00110000   ..**....
C2A6  DB  1C  ;00011100   ...***..
C2A7  DB  00  ;00000000   ........

C2A8  DB  00  ;00000000   ........
C2A9  DB  00  ;00000000   ........
C2AA  DB  66  ;01100110   .**..**.
C2AB  DB  66  ;01100110   .**..**.
C2AC  DB  66  ;01100110   .**..**.
C2AD  DB  66  ;01100110   .**..**.
C2AE  DB  3E  ;00111110   ..*****.
C2AF  DB  00  ;00000000   ........

C2B0  DB  00  ;00000000   ........
C2B1  DB  00  ;00000000   ........
C2B2  DB  66  ;01100110   .**..**.
C2B3  DB  66  ;01100110   .**..**.
C2B4  DB  66  ;01100110   .**..**.
C2B5  DB  3C  ;00111100   ..****..
C2B6  DB  18  ;00011000   ...**...
C2B7  DB  00  ;00000000   ........

C2B8  DB  00  ;00000000   ........
C2B9  DB  00  ;00000000   ........
C2BA  DB  63  ;01100011   .**...**
C2BB  DB  6B  ;01101011   .**.*.**
C2BC  DB  6B  ;01101011   .**.*.**
C2BD  DB  7F  ;01111111   .*******
C2BE  DB  36  ;00110110   ..**.**.
C2BF  DB  00  ;00000000   ........

C2C0  DB  00  ;00000000   ........
C2C1  DB  00  ;00000000   ........
C2C2  DB  66  ;01100110   .**..**.
C2C3  DB  3C  ;00111100   ..****..
C2C4  DB  18  ;00011000   ...**...
C2C5  DB  3C  ;00111100   ..****..
C2C6  DB  66  ;01100110   .**..**.
C2C7  DB  00  ;00000000   ........

C2C8  DB  00  ;00000000   ........
C2C9  DB  00  ;00000000   ........
C2CA  DB  66  ;01100110   .**..**.
C2CB  DB  66  ;01100110   .**..**.
```

```
C2CC  DB  66  ;01100110   .**..**.
C2CD  DB  3E  ;00111110   ..*****.
C2CE  DB  06  ;00000110   .....**.
C2CF  DB  3C  ;00111100   ..****..

C2D0  DB  00  ;00000000   ........
C2D1  DB  00  ;00000000   ........
C2D2  DB  7E  ;01111110   .******.
C2D3  DB  0C  ;00001100   ....**..
C2D4  DB  18  ;00011000   ...**...
C2D5  DB  30  ;00110000   ..**....
C2D6  DB  7E  ;01111110   .******.
C2D7  DB  00  ;00000000   ........

C2D8  DB  0C  ;00001100   ....**..
C2D9  DB  18  ;00011000   ...**...
C2DA  DB  18  ;00011000   ...**...
C2DB  DB  70  ;01110000   .***....
C2DC  DB  18  ;00011000   ...**...
C2DD  DB  18  ;00011000   ...**...
C2DE  DB  0C  ;00001100   ....**..
C2DF  DB  00  ;00000000   ........

C2E0  DB  18  ;00011000   ...**...
C2E1  DB  18  ;00011000   ...**...
C2E2  DB  18  ;00011000   ...**...
C2E3  DB  00  ;00000000   ........
C2E4  DB  18  ;00011000   ...**...
C2E5  DB  18  ;00011000   ...**...
C2E6  DB  18  ;00011000   ...**...
C2E7  DB  00  ;00000000   ........

C2E8  DB  30  ;00110000   ..**....
C2E9  DB  18  ;00011000   ...**...
C2EA  DB  18  ;00011000   ...**...
C2EB  DB  0E  ;00001110   ....***.
C2EC  DB  18  ;00011000   ...**...
C2ED  DB  18  ;00011000   ...**...
C2EE  DB  30  ;00110000   ..**....
C2EF  DB  00  ;00000000   ........

C2F0  DB  31  ;00110001   ..**...*
C2F1  DB  6B  ;01101011   .**.*.**
C2F2  DB  46  ;01000110   .*...**.
C2F3  DB  00  ;00000000   ........
C2F4  DB  00  ;00000000   ........
C2F5  DB  00  ;00000000   ........
C2F6  DB  00  ;00000000   ........
C2F7  DB  00  ;00000000   ........

C2F8  DB  FF  ;11111111   ********
C2F9  DB  FF  ;11111111   ********
C2FA  DB  FF  ;11111111   ********
C2FB  DB  FF  ;11111111   ********
```

```
C2FC  DB  FF  ;11111111   ********
C2FD  DB  FF  ;11111111   ********
C2FE  DB  FF  ;11111111   ********
C2FF  DB  FF  ;11111111   ********


C300 JMP  &CB1D

C303 DB  13
C304 DB  'BBC Computer '
C311 BRK

C312 DB '16K'
C315 DB 7 ;Bell
C316 BRK


C317 DB '32K'
C31A DB 7 ;Bell
C31B BRK

C31C DB 08,0D,0D


****** 4 COLOUR MODE BYTE MASK LOOK UP TABLE******


C31F  DB  00  ;00000000
C320  DB  11  ;00010001
C321  DB  22  ;00100010
C322  DB  33  ;00110011
C323  DB  44  ;01000100
C324  DB  55  ;01010101
C325  DB  66  ;01100110
C326  DB  77  ;01110111
C327  DB  88  ;10001000
C328  DB  99  ;10011001
C329  DB  AA  ;10101010
C32A  DB  BB  ;10111011
C32B  DB  CC  ;11001100
C32C  DB  DD  ;11011101
C32D  DB  EE  ;11101110
C32E  DB  FF  ;11111111


******16 COLOUR MODE BYTE MASK LOOK UP TABLE******


C32F  DB  00  ;00000000
C330  DB  55  ;01010101
C331  DB  AA  ;10101010
C332  DB  FF  ;11111111


****** VDU ENTRY POINT LO       LOOK UP TABLE******


C333  DB  11  ;00010001
```

```
C334  DB  3B  ;00111011
C335  DB  96  ;10010110
C336  DB  A1  ;10100001
C337  DB  AD  ;10101101
C338  DB  B9  ;10111001
C339  DB  11  ;00010001
C33A  DB  6F  ;01101111
C33B  DB  C5  ;11000101
C33C  DB  64  ;01100100
C33D  DB  F0  ;11110000
C33E  DB  5B  ;01011011
C33F  DB  59  ;01011001
C340  DB  AF  ;10101111
C341  DB  8D  ;10001101
C342  DB  A6  ;10100110
C343  DB  C0  ;11000000
C344  DB  F9  ;11111001
C345  DB  FD  ;11111101
C346  DB  92  ;10010010
C347  DB  39  ;00111001
C348  DB  9B  ;10011011
C349  DB  EB  ;11101011
C34A  DB  F1  ;11110001
C34B  DB  39  ;00111001
C34C  DB  8C  ;10001100
C34D  DB  BD  ;10111101
C34E  DB  11  ;00010001
C34F  DB  FA  ;11111010
C350  DB  A2  ;10100010
C351  DB  79  ;01111001
C352  DB  87  ;10000111
C353  DB  AC  ;10101100


****** VDU ENTRY POINT HI PARAMETER LOOK UP TABLE******


C354  DB  C5  ;11000101
C355  DB  2F  ;00101111
C356  DB  C5  ;11000101
C357  DB  C5  ;11000101
C358  DB  C5  ;11000101
C359  DB  C5  ;11000101
C35A  DB  C5  ;11000101
C35B  DB  E8  ;11101000
C35C  DB  C5  ;11000101
C35D  DB  C6  ;11000110
C35E  DB  C6  ;11000110
C35F  DB  C6  ;11000110
C360  DB  C7  ;11000111
C361  DB  C7  ;11000111
C362  DB  C5  ;11000101
C363  DB  C5  ;11000101
C364  DB  C7  ;11000111
C365  DB  4F  ;01001111
C366  DB  4E  ;01001110
C367  DB  5B  ;01011011
C368  DB  C8  ;11001000
C369  DB  C5  ;11000101
```

```
C36A  DB  5F  ;01011111
C36B  DB  57  ;01010111
C36C  DB  78  ;01111000
C36D  DB  6B  ;01101011
C36E  DB  C9  ;11001001
C36F  DB  C5  ;11000101
C370  DB  3C  ;00111100
C371  DB  7C  ;01111100
C372  DB  C7  ;11000111
C373  DB  4E  ;01001110
C374  DB  CA  ;11001010


****** *640 MULTIPLICATION TABLE  40 - 80 MODES ******


C375  DB  00  ;00000000
C376  DB  00  ;00000000
C377  DB  02  ;00000010
C378  DB  80  ;10000000
C379  DB  05  ;00000101
C37A  DB  00  ;00000000
C37B  DB  07  ;00000111
C37C  DB  80  ;10000000
C37D  DB  0A  ;00001010
C37E  DB  00  ;00000000
C37F  DB  0C  ;00001100
C380  DB  80  ;10000000
C381  DB  0F  ;00001111
C382  DB  00  ;00000000
C383  DB  11  ;00010001
C384  DB  80  ;10000000
C385  DB  14  ;00010100
C386  DB  00  ;00000000
C387  DB  16  ;00010110
C388  DB  80  ;10000000
C389  DB  19  ;00011001
C38A  DB  00  ;00000000
C38B  DB  1B  ;00011011
C38C  DB  80  ;10000000
C38D  DB  1E  ;00011110
C38E  DB  00  ;00000000
C38F  DB  20  ;00100000
C390  DB  80  ;10000000
C391  DB  23  ;00100011
C392  DB  00  ;00000000
C393  DB  25  ;00100101
C394  DB  80  ;10000000
C395  DB  28  ;00101000
C396  DB  00  ;00000000
C397  DB  2A  ;00101010
C398  DB  80  ;10000000
C399  DB  2D  ;00101101
C39A  DB  00  ;00000000
C39B  DB  2F  ;00101111
C39C  DB  80  ;10000000
C39D  DB  32  ;00110010
C39E  DB  00  ;00000000
C39F  DB  34  ;00110100
```

```
C3A0  DB  80  ;10000000
C3A1  DB  37  ;00110111
C3A2  DB  00  ;00000000
C3A3  DB  39  ;00111001
C3A4  DB  80  ;10000000
C3A5  DB  3C  ;00111100
C3A6  DB  00  ;00000000
C3A7  DB  3E  ;00111110
C3A8  DB  80  ;10000000
C3A9  DB  41  ;01000001
C3AA  DB  00  ;00000000
C3AB  DB  43  ;01000011
C3AC  DB  80  ;10000000
C3AD  DB  46  ;01000110
C3AE  DB  00  ;00000000
C3AF  DB  48  ;01001000
C3B0  DB  80  ;10000000
C3B1  DB  4B  ;01001011
C3B2  DB  00  ;00000000
C3B3  DB  4D  ;01001101


****** *40 MULTIPLICATION TABLE  TELETEXT  MODE ******


C3B4  DB  80  ;10000000
C3B5  DB  00  ;00000000
C3B6  DB  00  ;00000000
C3B7  DB  00  ;00000000
C3B8  DB  28  ;00101000
C3B9  DB  00  ;00000000
C3BA  DB  50  ;01010000
C3BB  DB  00  ;00000000
C3BC  DB  78  ;01111000
C3BD  DB  00  ;00000000
C3BE  DB  A0  ;10100000
C3BF  DB  00  ;00000000
C3C0  DB  C8  ;11001000
C3C1  DB  00  ;00000000
C3C2  DB  F0  ;11110000
C3C3  DB  01  ;00000001
C3C4  DB  18  ;00011000
C3C5  DB  01  ;00000001
C3C6  DB  40  ;01000000
C3C7  DB  01  ;00000001
C3C8  DB  68  ;01101000
C3C9  DB  01  ;00000001
C3CA  DB  90  ;10010000
C3CB  DB  01  ;00000001
C3CC  DB  B8  ;10111000
C3CD  DB  01  ;00000001
C3CE  DB  E0  ;11100000
C3CF  DB  02  ;00000010
C3D0  DB  08  ;00001000
C3D1  DB  02  ;00000010
C3D2  DB  30  ;00110000
C3D3  DB  02  ;00000010
C3D4  DB  58  ;01011000
C3D5  DB  02  ;00000010
```

```
C3D6  DB  80  ;10000000
C3D7  DB  02  ;00000010
C3D8  DB  A8  ;10101000
C3D9  DB  02  ;00000010
C3DA  DB  D0  ;11010000
C3DB  DB  02  ;00000010
C3DC  DB  F8  ;11111000
C3DD  DB  03  ;00000011
C3DE  DB  20  ;00100000
C3DF  DB  03  ;00000011
C3E0  DB  48  ;01001000
C3E1  DB  03  ;00000011
C3E2  DB  70  ;01110000
C3E3  DB  03  ;00000011
C3E4  DB  98  ;10011000
C3E5  DB  03  ;00000011


****** TEXT WINDOW -BOTTOM ROW LOOK UP TABLE ******


C3E6  DB  C0  ;11000000
C3E7  DB  1F  ;00011111
C3E8  DB  1F  ;00011111
C3E9  DB  1F  ;00011111
C3EA  DB  18  ;00011000
C3EB  DB  1F  ;00011111
C3EC  DB  1F  ;00011111
C3ED  DB  18  ;00011000
C3EE  DB  18  ;00011000


****** TEXT WINDOW -RIGHT HAND COLUMN LOOK UP TABLE ******


C3EF  DB  4F  ;01001111
C3F0  DB  27  ;00100111
C3F1  DB  13  ;00010011
C3F2  DB  4F  ;01001111
C3F3  DB  27  ;00100111
C3F4  DB  13  ;00010011
C3F5  DB  27  ;00100111
C3F6  DB  27  ;00100111




*************************************************************************
 *
*
 *       SEVERAL OF THE FOLLOWING TABLES OVERLAP EACH OTHER
*
 *       SOME ARE DUAL PURPOSE
*
 *
*
```

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



 ************** VIDEO ULA CONTROL REGISTER SETTINGS
************************

 C3F7  DB  9C  ;10011100
 C3F8  DB  D8  ;11011000
 C3F9  DB  F4  ;11110100
 C3FA  DB  9C  ;10011100
 C3FB  DB  88  ;10001000
 C3FC  DB  C4  ;11000100
 C3FD  DB  88  ;10001000
 C3FE  DB  4B  ;01001011


 ******** NUMBER OF BYTES PER CHARACTER FOR EACH DISPLAY MODE
************

 C3FF  DB  08  ;00001000
 C400  DB  10  ;00010000
 C401  DB  20  ;00100000
 C402  DB  08  ;00001000
 C403  DB  08  ;00001000
 C404  DB  10  ;00010000
 C405  DB  08  ;00001000
 C406  DB  01  ;00000001



 ******************** MASK TABLE FOR  2 COLOUR MODES
**********************

 C407  DB  AA  ;10101010
 C408  DB  55  ;01010101



 ****************** MASK TABLE FOR  4 COLOUR MODES
************************

 C409  DB  88  ;10001000
 C40A  DB  44  ;01000100
 C40B  DB  22  ;00100010
 C40C  DB  11  ;00010001
```

```
 ********** MASK TABLE FOR  4 COLOUR MODES FONT FLAG MASK TABLE
**********


 C40D  DB  80  ;10000000
 C40E  DB  40  ;01000000
 C40F  DB  20  ;00100000
 C410  DB  10  ;00010000
 C411  DB  08  ;00001000
 C412  DB  04  ;00000100
 C413  DB  02  ;00000010
 C414  DB  01  ;00000001


 ********* NUMBER OF COLOURS -1 FOR EACH MODE
****************************


 C414  DB  01  ;00000001
 C415  DB  03  ;00000011
 C416  DB  0F  ;00001111
 C417  DB  01  ;00000001
 C418  DB  01  ;00000001
 C419  DB  03  ;00000011
 C41A  DB  01  ;00000001
 C41B  DB  00  ;00000000



 ************** GCOL PLOT OPTIONS PROCESSING LOOK UP TABLE
***************


 C41C  DB  FF  ;11111111
 C41D  DB  00  ;00000000
 C41E  DB  00  ;00000000
 C41F  DB  FF  ;11111111
 C420  DB  FF  ;11111111
 C421  DB  FF  ;11111111
 C422  DB  FF  ;11111111
 C423  DB  00  ;00000000
 C424  DB  00  ;00000000
 C425  DB  FF  ;11111111



 *************** 2 COLOUR MODES PARAMETER LOOK UP TABLE
******************


 C424  DB  00  ;00000000
 C425  DB  FF  ;11111111
```

```
 *************** 4 COLOUR MODES PARAMETER LOOK UP TABLE
******************


 C426  DB  00  ;00000000
 C427  DB  0F  ;00001111
 C428  DB  F0  ;11110000
 C429  DB  FF  ;11111111



 ***************16 COLOUR MODES PARAMETER LOOK UP TABLE
******************


 C42A  DB  00  ;00000000
 C42B  DB  03  ;00000011
 C42C  DB  0C  ;00001100
 C42D  DB  0F  ;00001111
 C42E  DB  30  ;00110000
 C42F  DB  33  ;00110011
 C430  DB  3C  ;00111100
 C431  DB  3F  ;00111111
 C432  DB  C0  ;11000000
 C433  DB  C3  ;11000011
 C434  DB  CC  ;11001100
 C435  DB  CF  ;11001111
 C436  DB  F0  ;11110000
 C437  DB  F3  ;11110011
 C438  DB  FC  ;11111100
 C439  DB  FF  ;11111111



 ********** DISPLAY MODE PIXELS/BYTE-1 LOOK UP TABLE
**********************


 C43A  DB  07  ;00000111
 C43B  DB  03  ;00000011
 C43C  DB  01  ;00000001
 C43D  DB  00  ;00000000
 C43E  DB  07  ;00000111
 C43F  DB  03  ;00000011
 C440  DB  00  ;00000000
 C441  DB  00  ;00000000



 ************ SCREEN DISPLAY MEMORY INDEX LOOK UP TABLE
******************


 C440  DB  00  ;00000000
 C441  DB  00  ;00000000
 C442  DB  00  ;00000000
 C443  DB  01  ;00000001
```

```
 C444  DB  02  ;00000010
 C445  DB  02  ;00000010
 C446  DB  03  ;00000011
 C447  DB  04  ;00000100



 *********** SOUND PITCH OFFSET BY CHANNEL LOOK UP TABLE
*****************


 C441  DB  00  ;00000000
 C442  DB  00  ;00000000
 C443  DB  01  ;00000001
 C444  DB  02  ;00000010



 ***************** CRTC SET UP PARAMETERS TABLE 1
************************


 C44B  DB  0D  ;00001101
 C44C  DB  05  ;00000101
 C44D  DB  0D  ;00001101
 C44E  DB  05  ;00000101



 ***************** CRTC SET UP PARAMETERS TABLE 2
************************


 C44F  DB  04  ;00000100
 C450  DB  04  ;00000100
 C451  DB  0C  ;00001100
 C452  DB  0C  ;00001100
 C453  DB  04  ;00000100



 ***************** VDU SECTION CONTROL NUMBERS
***************************


 C447  DB  04  ;00000100
 C448  DB  00  ;00000000
 C449  DB  06  ;00000110
 C44A  DB  02  ;00000010
 C44B  DB  0D  ;00001101
 C44C  DB  05  ;00000101
 C44D  DB  0D  ;00001101
 C44E  DB  05  ;00000101
 C44F  DB  04  ;00000100
 C450  DB  04  ;00000100
 C451  DB  0C  ;00001100
```

```
 C452  DB  0C  ;00001100
 C453  DB  04  ;00000100
 C454  DB  02  ;00000010
 C455  DB  32  ;00110010
 C456  DB  7A  ;01111010
 C457  DB  92  ;10010010
 C458  DB  E6  ;11100110



 ************** MSB OF MEMORY OCCUPIED BY SCREEN BUFFER
*****************


 C459  DB  50  ;01010000
 C45A  DB  40  ;01000000
 C45B  DB  28  ;00101000
 C45C  DB  20  ;00100000
 C45D  DB  04  ;00000100



 ************ MSB OF FIRST LOCATION OCCUPIED BY SCREEN BUFFER
************


 C45E  DB  30  ;00110000
 C45F  DB  40  ;01000000
 C460  DB  58  ;01011000
 C461  DB  60  ;01100000
 C462  DB  7C  ;01111100



 ****************** NUMBER OF BYTES PER ROW
*******************************


 C463  DB  28  ;00101000
 C464  DB  40  ;01000000
 C465  DB  80  ;10000000



 ******** ROW MULTIPLIACTION TABLE POINTER TO LOOK UP TABLE
**************


 C466  DB  B5  ;10110101
 C467  DB  75  ;01110101
 C468  DB  75  ;01110101



 ********** CRTC CURSOR END REGISTER SETTING LOOK UP TABLE
***************
```

```
 C469  DB  0B  ;00001011
 C46A  DB  17  ;00010111
 C46B  DB  23  ;00100011
 C46C  DB  2F  ;00101111
 C46D  DB  3B  ;00111011



 ************* 6845 REGISTERS 0-11 FOR MODES 0-2
*************************

 C46E  DB  7F  ;01111111
 C46F  DB  50  ;01010000
 C470  DB  62  ;01100010
 C471  DB  28  ;00101000
 C472  DB  26  ;00100110
 C473  DB  00  ;00000000
 C474  DB  20  ;00100000
 C475  DB  22  ;00100010
 C476  DB  01  ;00000001
 C477  DB  07  ;00000111
 C478  DB  67  ;01100111
 C479  DB  08  ;00001000



 ************* 6845 REGISTERS 0-11 FOR MODE 3
*****************************

 C47A  DB  7F  ;01111111
 C47B  DB  50  ;01010000
 C47C  DB  62  ;01100010
 C47D  DB  28  ;00101000
 C47E  DB  1E  ;00011110
 C47F  DB  02  ;00000010
 C480  DB  19  ;00011001
 C481  DB  1B  ;00011011
 C482  DB  01  ;00000001
 C483  DB  09  ;00001001
 C484  DB  67  ;01100111
 C485  DB  09  ;00001001



 ************ 6845 REGISTERS 0-11 FOR MODES 4-5
*************************

 C486  DB  3F  ;00111111
 C487  DB  28  ;00101000
 C488  DB  31  ;00110001
 C489  DB  24  ;00100100
 C48A  DB  26  ;00100110
```

```
 C48B  DB  00  ;00000000
 C48C  DB  20  ;00100000
 C48D  DB  22  ;00100010
 C48E  DB  01  ;00000001
 C48F  DB  07  ;00000111
 C490  DB  67  ;01100111
 C491  DB  08  ;00001000



 ********** 6845 REGISTERS 0-11 FOR MODE 6
*******************************


 C492  DB  3F  ;00111111
 C493  DB  28  ;00101000
 C494  DB  31  ;00110001
 C495  DB  24  ;00100100
 C496  DB  1E  ;00011110
 C497  DB  02  ;00000010
 C498  DB  19  ;00011001
 C499  DB  1B  ;00011011
 C49A  DB  01  ;00000001
 C49B  DB  09  ;00001001
 C49C  DB  67  ;01100111
 C49D  DB  09  ;00001001


 ********* 6845 REGISTERS 0-11 FOR MODE 7
*******************************


 C49E  DB  3F  ;00111111
 C49F  DB  28  ;00101000
 C4A0  DB  33  ;00110011
 C4A1  DB  24  ;00100100
 C4A2  DB  1E  ;00011110
 C4A3  DB  02  ;00000010
 C4A4  DB  19  ;00011001
 C4A5  DB  1B  ;00011011
 C4A6  DB  93  ;10010011
 C4A7  DB  12  ;00010010
 C4A8  DB  72  ;01110010
 C4A9  DB  13  ;00010011



 ************* VDU ROTINE VECTOR ADDRESSES
*******************************


 C4AA  DB  86  ;10000110
 C4AB  DB  D3  ;11010011
 C4AC  DB  7E  ;01111110
 C4AD  DB  D3  ;11010011
```

```
 ************ VDU ROUTINE BRANCH VECTOR ADDRESS LO
************************

 C4AE  DB  6A  ;01101010
 C4AF  DB  74  ;01110100
 C4B0  DB  42  ;01000010
 C4B1  DB  4B  ;01001011



 ************ VDU ROUTINE BRANCH VECTOR ADDRESS HI
************************

 C4B2  DB  D3  ;11010011
 C4B3  DB  D3  ;11010011
 C4B4  DB  D3  ;11010011
 C4B5  DB  D3  ;11010011



 *********** TELETEXT CHARACTER CONVERSION TABLE
*************************

 C4B6  DB  23  ;00100011
 C4B7  DB  5F  ;01011111
 C4B8  DB  60  ;01100000
 C4B9  DB  23  ;00100011



 *********** SOFT CHARACTER RAM ALLOCATION
*****************************

 C4BA  DB  04  ;00000100
 C4BB  DB  05  ;00000101
 C4BC  DB  06  ;00000110
 C4BD  DB  00  ;00000000
 C4BE  DB  01  ;00000001
 C4BF  DB  02  ;00000010



*******************************************************************************
 *
*
 *
*
 *       VDU FUNCTIONS ADDRESSES
*
```

```
 *
*
 *
*
```

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

```
         ; VDU   Address      Parameters         function

         ;  0    &C511           0       does nothing
         ;  1    &C53B           1       next character to printer only
         ;  2    &C596           0       enable printer
         ;  3    &C5A1           0       disable printer
         ;  4    &C5AD           0       select text cursor
         ;  5    &C5B9           0       select graphics cursor
         ;  6    &C511           0       enable display
         ;  7    &E86F           0       bell
         ;  8    &C5C5           0       cursor left
         ;  9    &C664           0       cursor right
         ; 10    &C6F0           0       cursor down
         ; 11    &C65B           0       cursor up
         ; 12    &C759           0       clear text window
         ; 13    &C7AF           0       newline
         ; 14    &C58D           0       select paged mode
         ; 15    &C5A6           0       cancel paged mode
         ; 16    &C7C0           0       clear graphics screen
         ; 17    &C7F9           1       define text colour
         ; 18    &C7FD           2       define graphics colour
         ; 19    &C892           5       define logical colour
         ; 20    &C839           0       restore default colours
         ; 21    &C59B           0       disable display
         ; 22    &C8EB           1       select screen MODE
         ; 23    &C8F1           9       define character
         ; 24    &CA39           8       define graphics window
         ; 25    &C98C           5       PLOT
         ; 26    &C9BD           0       set default windows
         ; 27    &C511           0       ESCAPE (does nothing)
         ; 28    &C6FA           4       define text window
         ; 29    &CAA2           4       define graphics origin
         ; 30    &C779           0       home cursor
         ; 31    &C787           2       position text cursor (TAB)
         ;127    &CAAC           0       delete
```

```
*************************************************************************
 *
*
 *
*
 *      VDU Variables
*
 *
*
 *
*
```

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

```
         ;D0     VDU status
         ;Bit    0       printer output enabled
```

```
;       1       scrolling disabled
;       2       paged scrolling enabled
;       3       software scrolling selected
;       4       not used
;       5       printing at graphics cursor enabled
;       6       cursor editing mode enabled
;       7       screen disabled

;D1     byte mask for current graphics point
;D2/3   text colour bytes to be ORed and EORed into memory
;D4/5   graphics colour bytes to be ORed and EORed into memory
;D6/7   address of top line of current graphics cell
;D8/9   address of top scan line of current text character
;DA/F   temporary workspace
;E0/1   CRTC row multiplication table pointer


;246    Character definition explosion switch

;248    current video ULA control regiter setting
;249    current pallette setting

;251    flash counter
;252    mark-space count
;253    space period count

;256    EXEC file handle
;257    SPOOL file handle

;260    Econet OSWRCH interception flag
;267    bit 7 set ignore start up message
;268    length of key string
;269    print line counter
;26A    number of items in VDU queque
;26B    TAB key value
;26C    ESCAPE character

;27D    cursor editing status

;28F    start up options (Keyboard links)
       bits    0-2     default screen Mode
                 3     reverse SHIFT/BREAK
               4-5     disc timing parameters
;290    screen display vertical adjustment
;291    interlace toggle flag

;300/1  graphics window left
;302/3  graphics window bottom
;304/5  graphics window right
;306/7  graphics window top
;308    text window left
;309    text window bottom
;30A    text window right
;30B    text window top
;30C/D  graphics origin, horizontal (external values)
;30E/F  graphics origin, vertical   (external values)
;310/1  current graphics cursor, horizontal (external values)
;312/3  current graphics cursor, vertical   (external values)
;314/5  last graphics cursor, horizontal    (external values)
;316/7  last graphics cursor, vertical      (external values)
```

```
;318    text column
;319    text line
;31A    graphics scan line expressed as line of character
;31B-323 VDU parameters, last parameter in &323
;324/5  current graphics cursor, horizontal (internal values)
;316/7  current graphics cursor, vertical   (internal values)
;328-349 general workspace
;34A/B  text cursor address to CRT controller
;34C/D  width of text window in bytes
;34E    hi byte of address of screen RAM start
;34F    bytes per character
;350/1  address of window area start
;352/3  bytes per character row
;354    high byte of screen RAM size
;355    Mode
;356    memory map type
;357/35A current colours
;35B/C  graphics plot mode
;35D/E  jump vector
;35F    last setting of CRT controller Cursor start register
;360    number of logical colours less 1
;361    pixels per byte (0  in text only modes)
;362/3  colour masks
;364/5  X/Y for text input cursor
;366    output cursor character for MODE 7
;367    Font flag
;368/E  font location bytes
;36F-37E Colour palette
```

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

*********************************************************************
*
 **
**
 **      OSWRCH  MAIN ROUTINE  entry from E0C5
**
 **
**
 **      output a byte via the VDU stream
**
 **
**

*********************************************************************
*

*********************************************************************
*
 ;this routine takes up over 40% of the operating system ROM
 ;entry points are variable, as are the results achieved.
 ;tracing any particular path is relatively easy but generalising for
 ;commenting is not.  For clarity comments will not be as detailed as
 ;for later parts of the Operating system.

 C4C0    LDX     &026A   ;get number of items in VDU queque
 C4C3    BNE     &C512   ;if parameters needed then C512
 C4C5    BIT     &D0     ;else check status byte
 C4C7    BVC     &C4D8   ;if cursor editing enabled two cursors exist
 C4C9    JSR     &C568   ;swap values
 C4CC    JSR     &CD6A   ;then set up write cursor
 C4CF    BMI     &C4D8   ;if display disabled C4D8
 C4D1    CMP     #&0D    ;else if character in A=RETURN teminate edit
 C4D3    BNE     &C4D8   ;else C4D8

 C4D5    JSR     &D918   ;terminate edit

 C4D8    CMP     #&7F    ;is character DELETE ?
 C4DA    BEQ     &C4ED   ;if so C4ED

 C4DC    CMP     #&20    ;is it less than space? (i.e. VDU control code)
 C4DE    BCC     &C4EF   ;if so C4EF
 C4E0    BIT     &D0     ;else check VDU byte ahain
 C4E2    BMI     &C4EA   ;if screen disabled C4EA
 C4E4    JSR     &CFB7   ;else display a character
 C4E7    JSR     &C664   ;and cursor right
 C4EA    JMP     &C55E   ;

 ********* read link addresses and number of parameters *****************

 C4ED    LDA     #&20    ;to replace delete character

 ********* read link addresses and number of parameters *****************

 C4EF    TAY             ;Y=A
 C4F0    LDA     &C333,Y ;get lo byte of link address
 C4F3    STA     &035D   ;store it in jump vector
```

```
 C4F6    LDA     &C354,Y ;get hi byte
 C4F9    BMI     &C545   ;if negative (as it will be if a direct address)
                         ;there are no parameters needed
                         ;so C545
 C4FB    TAX             ;else X=A
 C4FC    ORA     #&F0    ;set up negated parameter count
 C4FE    STA     &026A   ;store it as number of items in VDU queue
 C501    TXA             ;get back A
 C502    LSR             ;A=A/16
 C503    LSR             ;
 C504    LSR             ;
 C505    LSR             ;
 C506    CLC             ;clear carry
 C507    ADC     #&C3    ;add &C3 to get hi byte of link address
 C509    STA     &035E   ;
 C50C    BIT     &D0     ;check if cursor editing enabled
 C50E    BVS     &C52F   ;if so re-exchange pointers
 C510    CLC             ;clear carry
 C511    RTS             ;and exit

 ;return with carry clear indicates that printer action not required.

 ;
 ********** parameters are outstanding
**********************************
 X=&26A = 2 complement of number of parameters X=&FF for 1, FE for 2 etc.

 C512    STA     &0224,X ;store parameter in queue
 C515    INX             ;increment X
 C516    STX     &026A   ;store it as VDU queue
 C519    BNE     &C532   ;if not 0 C532 as more parameters are needed
 C51B    BIT     &D0     ;get VDU status byte
 C51D    BMI     &C534   ;if screen disabled C534
 C51F    BVS     &C526   ;else if cursor editing C526
 C521    JSR     &CCF5   ;execute required function
 C524    CLC             ;clear carry
 C525    RTS             ;and exit
 ;
 C526    JSR     &C568   ;swap values of cursors
 C529    JSR     &CD6A   ;set up write cursor
 C52C    JSR     &CCF5   ;execute required function
 C52F    JSR     &C565   ;re-exchange pointers

 C532    CLC             ;carry clear
 C533    RTS             ;exit
```

```
*************************************************************************
 *
*
 *       VDU 1 send next character to printer only
*
 *
*
 *       1 parameter required
*
```

```
 *
*

*********************************************************************
 ;
 C534    LDY     &035E   ;if upper byte of link address not &C5
 C537    CPY     #&C5    ;printer is not interested
 C539    BNE     &C532   ;so C532
 C53B    TAX             ;else X=A
 C53C    LDA     &D0     ;A=VDU status byte
 C53E    LSR             ;get bit 0 into carry
 C53F    BCC     &C511   ;if printer not enabled exit
 C541    TXA             ;restore A
 C542    JMP     &E11E   ;else send byte in A (next byte) to printer



 *********** if explicit link address found, no parameters
***************

 C545    STA     &035E   ;upper byte of link address
 C548    TYA             ;restore A
 C549    CMP     #&08    ;is it 7 or less?
 C54B    BCC     &C553   ;if so C553
 C54D    EOR     #&FF    ;invert it
 C54F    CMP     #&F2    ;c is set if A >&0D
 C551    EOR     #&FF    ;re invert

 C553    BIT     &D0     ;VDU status byte
 C555    BMI     &C580   ;if display disabled C580
 C557    PHP             ;push processor flags
 C558    JSR     &CCF5   ;execute required function
 C55B    PLP             ;get back flags
 C55C    BCC     &C561   ;if carry clear (from C54B/F)



 **************** main exit routine
************************************

 C55E    LDA     &D0     ;VDU status byte
 C560    LSR             ;Carry is set if printer is enabled
 C561    BIT     &D0     ;VDU status byte
 C563    BVC     &C511   ;if nmo cursor editing  C511 to exit


 ***************** cursor editing routines
******************************

 C565    JSR     &CD7A   ;restore normal write cursor

 C568    PHP             ;save flags and
 C569    PHA             ;A
 C56A    LDX     #&18    ;X=&18
 C56C    LDY     #&64    ;Y=&64
 C56E    JSR     &CDDE   ;exchange &300/1+X with &300/1+Y
 C571    JSR     &CF06   ;set up display address
 C574    JSR     &CA02   ;set cursor position
 C577    LDA     &D0     ;VDU status byte
 C579    EOR     #&02    ;invert bit 1 to allow or bar scrolling
```

```
C57B    STA     &D0     ;VDU status byte
C57D    PLA             ;restore flags and A
C57E    PLP             ;
C57F    RTS             ;and exit
;
C580    EOR     #&06    ;if A<>6
C582    BNE     &C58C   ;return via C58C
C584    LDA     #&7F    ;A=&7F
C586    BCC     &C5A8   ;and goto C5A8 ALWAYS!!


****************** check text cursor in use ***************************

C588    LDA     &D0     ;VDU status byte
C58A    AND     #&20    ;set A from bit 5 of status byte
C58C    RTS             ;and exit

A=0 if text cursor, &20 if graphics



*******************************************************************************
 *
*
 *
*
 *       SET PAGED MODE  VDU 14
*
 *
*
 *
*

*******************************************************************************

 ;
C58D    LDY     #&00    ;Y=0
C58F    STY     &0269   ;paged mode counter
C592    LDA     #&04    ;A=04
C594    BNE     &C59D   ;jump to C59D




*******************************************************************************
 *
*
 *
*
 *       VDU 2 PRINTER ON
*
 *
*
 *
*

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

```
 C596    JSR     &E1A2   ;select printer buffer  and output character
 C599    LDA     #&94    ;A=&94
                         ;when inverted at C59B this =1



*************************************************************************
 *
*
 *
*
 *       DISABLE DISPLAY VDU 21
*
 *       no parameters
*
 *
*

*************************************************************************


 C59B    EOR     #&95    ;if A=&15 A now =&80: if A=&94 A now =1

 C59D    ORA     &D0     ;VDU status byte set bit 0 or bit 7
 C59F    BNE     &C5AA   ;




*************************************************************************
 *
*
 *       VDU 3 Printer off
*
 *
*
 *       No parameters
*
 *
*

*************************************************************************


 C5A1    JSR     &E1A2   ;select printer buffer  and output character
 C5A4    LDA     #&0A    ;A=10




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

```
 *       VDU 15 paged mode off   No parameters
*
 *
*
 *
*

*********************************************************************
 A=&F or &A

 C5A6    EOR     #&F4    ;convert to &FB or &FE
 C5A8    AND     &D0     ;VDU status byte clear bit 0 or bit 2 of status
 C5AA    STA     &D0     ;VDU status byte
 C5AC    RTS             ;exit




*********************************************************************
 *
*
 *
*
 *       VDU 4 select Text Cursor  No parameters
*
 *
*
 *
*

*********************************************************************
 ;
 C5AD    LDA     &0361   ;pixels per byte
 C5B0    BEQ     &C5AC   ;if no graphics in current mode C5AC
 C5B2    JSR     &C951   ;set CRT controller for text cursor
 C5B5    LDA     #&DF    ;this to clear bit 5 of status byte
 C5B7    BNE     &C5A8   ;via C5A8 exit




*********************************************************************
 *
*
 *
*
 *       VDU 5 set graphics cursor
*
 *
*
 *
*

*********************************************************************

 C5B9    LDA     &0361   ;pixels per byte
 C5BC    BEQ     &C5AC   ;if none this is text mode so exit
```

```
 C5BE    LDA     #&20    ;set up graphics cursor
 C5C0    JSR     &C954   ;via C954
 C5C3    BNE     &C59D   ;set bit 5 via exit C59D



*************************************************************************
 *
*
 *
*
 *       VDU 8  CURSOR LEFT     NO PARAMETERS
*
 *
*
 *
*

*************************************************************************


 C5C5    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C5C8    BNE     &C61F   ;move cursor left 8 pixels if graphics
 C5CA    DEC     &0318   ;else decrement text column
 C5CD    LDX     &0318   ;store new text column
 C5D0    CPX     &0308   ;if it is less than text window left
 C5D3    BMI     &C5EE   ;do wraparound  cursor to rt of screen 1 line up
 C5D5    LDA     &034A   ;text cursor 6845 address
 C5D8    SEC             ;subtract
 C5D9    SBC     &034F   ;bytes per character
 C5DC    TAX             ;put in X
 C5DD    LDA     &034B   ;get text cursor 6845 address
 C5E0    SBC     #&00    ;subtract 0
 C5E2    CMP     &034E   ;compare with hi byte of screen RAM address
 C5E5    BCS     &C5EA   ;if = or greater
 C5E7    ADC     &0354   ;add screen RAM size hi byte to wrap around
 C5EA    TAY             ;Y=A
 C5EB    JMP     &C9F6   ;Y hi and X lo byte of cursor position


 ***************** execute wraparound left-
up*****************************

 C5EE    LDA     &030A   ;text window right
 C5F1    STA     &0318   ;text column


 *************** cursor up
*********************************************

 C5F4    DEC     &0269   ;paged mode counter
 C5F7    BPL     &C5FC   ;if still greater than 0 skip next instruction
 C5F9    INC     &0269   ;paged mode counter to restore X=0
 C5FC    LDX     &0319   ;current text line
 C5FF    CPX     &030B   ;top of text window
 C602    BEQ     &C60A   ;if its at top of window C60A
```

```
 C604    DEC     &0319   ;else decrement current text line
 C607    JMP     &C6AF   ;and carry on moving cursor


 ******** cursor at top of window
**************************************

 C60A    CLC             ;clear carry
 C60B    JSR     &CD3F   ;check for window violatations
 C60E    LDA     #&08    ;A=8 to check for software scrolling
 C610    BIT     &D0     ;compare against VDU status byte
 C612    BNE     &C619   ;if not enabled C619
 C614    JSR     &C994   ;set screen start register and adjust RAM
 C617    BNE     &C61C   ;jump C61C

 C619    JSR     &CDA4   ;soft scroll 1 line
 C61C    JMP     &C6AC   ;and exit



 **********cursor left and down with graphics cursor in use
**************

 C61F    LDX     #&00    ;X=0 to select horizontal parameters


 ********** cursor down with graphics in use
*****************************
 ;X=2 for vertical or 0 for horizontal

 C621    STX     &DB     ;store X
 C623    JSR     &D10D   ;check for window violations
 C626    LDX     &DB     ;restore X
 C628    SEC             ;set carry
 C629    LDA     &0324,X ;current graphics cursor X>1=vertical
 C62C    SBC     #&08    ;subtract 8 to move back 1 character
 C62E    STA     &0324,X ;store in current graphics cursor X>1=verticaal
 C631    BCS     &C636   ;if carry set skip next
 C633    DEC     &0325,X ;current graphics cursor hi -1
 C636    LDA     &DA     ;&DA=0 if no violation else 1 if vert violation
                         ;2 if horizontal violation
 C638    BNE     &C658   ;if violation C658
 C63A    JSR     &D10D   ;check for window violations
 C63D    BEQ     &C658   ;if none C658

 C63F    LDX     &DB     ;else get back X
 C641    LDA     &0304,X ;graphics window rt X=0 top X=2
 C644    CPX     #&01    ;is X=0
 C646    BCS     &C64A   ;if not C64A
 C648    SBC     #&06    ;else subtract 7

 C64A    STA     &0324,X ;current graphics cursor X>1=vertical
 C64D    LDA     &0305,X ;graphics window hi rt X=0 top X=2
 C650    SBC     #&00    ;subtract carry
 C652    STA     &0325,X ;current graphics cursor X<2=horizontal else
vertical
 C655    TXA             ;A=X
 C656    BEQ     &C660   ;cursor up
 C658    JMP     &D1B8   ;set up external coordinates for graphics
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 11 Cursor Up    No Parameters
*
 *
*
 *
*

*************************************************************************

 C65B    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C65E    BEQ     &C5F4   ;if text cursor then C5F4
 C660    LDX     #&02    ;else X=2
 C662    BNE     &C6B6   ;goto C6B6




*************************************************************************
 *
*
 *
*
 *       VDU 9 Cursor right    No parameters
*
 *
*
 *
*

*************************************************************************

 C664    LDA     &D0     ;VDU status byte
 C666    AND     #&20    ;check bit 5
 C668    BNE     &C6B4   ;if set then graphics cursor in use so C6B4
 C66A    LDX     &0318   ;text column
 C66D    CPX     &030A   ;text window right
 C670    BCS     &C684   ;if X exceeds window right then C684
 C672    INC     &0318   ;text column
 C675    LDA     &034A   ;text cursor 6845 address
 C678    ADC     &034F   ;add bytes per character
 C67B    TAX             ;X=A
 C67C    LDA     &034B   ;text cursor 6845 address
 C67F    ADC     #&00    ;add carry if set
 C681    JMP     &C9F6   ;use X and Y to set new cursor address

 ********: text cursor down and right
************************************
```

```
 C684    LDA     &0308   ;text window left
 C687    STA     &0318   ;text column

 ********: text cursor down ************************************

 C68A    CLC             ;clear carry
 C68B    JSR     &CAE3   ;check bottom margin, X=line count
 C68E    LDX     &0319   ;current text line
 C691    CPX     &0309   ;bottom margin
 C694    BCS     &C69B   ;if X=>current bottom margin C69B
 C696    INC     &0319   ;else increment current text line
 C699    BCC     &C6AF   ;
 C69B    JSR     &CD3F   ;check for window violations
 C69E    LDA     #&08    ;check bit 3
 C6A0    BIT     &D0     ;VDU status byte
 C6A2    BNE     &C6A9   ;if software scrolling enabled C6A9
 C6A4    JSR     &C9A4   ;perform hardware scroll
 C6A7    BNE     &C6AC   ;
 C6A9    JSR     &CDFF   ;execute upward scroll
 C6AC    JSR     &CEAC   ;clear a line

 C6AF    JSR     &CF06   ;set up display address
 C6B2    BCC     &C732   ;


 *********** graphic cursor right
**************************************

 C6B4    LDX     #&00    ;


 ************** graphic cursor up  (X=2)
**********************************

 C6B6    STX     &DB     ;store X
 C6B8    JSR     &D10D   ;check for window violations
 C6BB    LDX     &DB     ;get back X
 C6BD    CLC             ;clear carry
 C6BE    LDA     &0324,X ;current graphics cursor X>1=vertical
 C6C1    ADC     #&08    ;Add 8 pixels
 C6C3    STA     &0324,X ;current graphics cursor X>1=vertical
 C6C6    BCC     &C6CB   ;
 C6C8    INC     &0325,X ;current graphics cursor X<2=horizontal else
vertical
 C6CB    LDA     &DA     ;A=0 no window violations 1 or 2 indicates
violation
 C6CD    BNE     &C658   ;if outside window C658
 C6CF    JSR     &D10D   ;check for window violations
 C6D2    BEQ     &C658   ;if no violations C658

 C6D4    LDX     &DB     ;get back X
 C6D6    LDA     &0300,X ;graphics window X<2 =left else bottom
 C6D9    CPX     #&01    ;If X=0
 C6DB    BCC     &C6DF   ;C6DF
 C6DD    ADC     #&06    ;else add 7
 C6DF    STA     &0324,X ;current graphics cursor X>1=vertical
 C6E2    LDA     &0301,X ;graphics window hi X<2 =left else bottom
 C6E5    ADC     #&00    ;add anny carry
 C6E7    STA     &0325,X ;current graphics cursor X<2=horizontal else
vertical
```

```
 C6EA    TXA             ;A=X
 C6EB    BEQ     &C6F5   ;if X=0 C6F5 cursor down
 C6ED    JMP     &D1B8   ;set up external coordinates for graphics
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 10  Cursor down     No parameters
*
 *
*
 *
*
```

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

 C6F0    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C6F3    BEQ     &C68A   ;if text cursor back to C68A
 C6F5    LDX     #&02    ;else X=2 to indicate vertical movement
 C6F7    JMP     &C621   ;move graphics cursor down
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 28   define text window         4 parameters
*
 *
*
 *
*
```

```
*************************************************************************
 ;parameters are set up thus
 ;0320  P1 left margin
 ;0321  P2 bottom margin
 ;0322  P3 right margin
 ;0323  P4 top margin
 ;Note that last parameter is always in 0323

 C6FA    LDX     &0355   ;screen mode
 C6FD    LDA     &0321   ;get bottom margin
 C700    CMP     &0323   ;compare with top margin
 C703    BCC     &C758   ;if bottom margin exceeds top return
 C705    CMP     &C3E7,X ;text window bottom margin maximum
 C708    BEQ     &C70C   ;if equal then its OK
 C70A    BCS     &C758   ;else exit
```

```
 C70C    LDA     &0322   ;get right margin
 C70F    TAY             ;put it in Y
 C710    CMP     C3EF,X  ;text window right hand margin maximum
 C713    BEQ     &C717   ;if equal then OK
 C715    BCS     &C758   ;if greater than maximum exit

 C717    SEC             ;set carry to subtract
 C718    SBC     &0320   ;left margin
 C71B    BMI     &C758   ;if left greater than right exit
 C71D    TAY             ;else A=Y (window width)
 C71E    JSR     &CA88   ;calculate number of bytes in a line
 C721    LDA     #&08    ;A=8 to set bit  of &D0
 C723    JSR     &C59D   ;indicating that text window is defined
 C726    LDX     #&20    ;point to parameters
 C728    LDY     #&08    ;point to text window margins
 C72A    JSR     &D48A   ;(&300/3+Y)=(&300/3+X)
 C72D    JSR     &CEE8   ;set up screen address
 C730    BCS     &C779   ;home cursor within window
 C732    JMP     &CA02   ;set cursor position




*************************************************************************
 *
*
 *
*
 *       OSWORD 9    read a pixel
*
 *
*
 *
*

*************************************************************************
 ;on entry &EF=A=9
 ;         &F0=X=low byte of parameter block address
 ;         &F1=Y=high byte of parameter block address
 ;       PARAMETER BLOCK
 ;bytes 0,1 X coordinate, bytes 2,3 Y coordinate
 ;EXIT with result in byte 4 =&FF if point was of screen or logical
colour
 ;     of point if on screen

 C735    LDY     #&03    ;Y=3 to point to hi byte of Y coordinate
 C737    LDA     (&F0),Y ;get it
 C739    STA     &0328,Y ;store it
 C73C    DEY             ;point to next byte
 C73D    BPL     &C737   ;transfer till Y=&FF lo byte of X coordinate in
&328
 C73F    LDA     #&28    ;
 C741    JSR     &D839   ;check window boundaries
 C744    LDY     #&04    ;Y=4
 C746    BNE     &C750   ;jump to C750
```

```
*************************************************************************
 *
*
 *
*
 *       OSWORD 11       read pallette
*
 *
*
 *
*

*************************************************************************
 ;on entry &EF=A=11
 ;         &F0=X=low byte of parameter block address
 ;         &F1=Y=high byte of parameter block address
 ;       PARAMETER BLOCK
 ;bytes 0,logical colour to read
 ;EXIT with result in  4 bytes:-0 logical colour,1 physical colour
 ;     2,3 both 0.  corresponds to reading VDU 19

 C748    AND     &0360   ;number of logical colours less 1
 C74B    TAX             ;put it in X
 C74C    LDA     &036F,X ;colour pallette
 C74F    INY             ;increment Y to point to byte 1

 C750    STA     (&F0),Y ;store data
 C752    LDA     #&00    ;issue 0s
 C754    CPY     #&04    ;to next bytes until Y=4
 C756    BNE     &C74F   ;

 C758    RTS             ;and exit




*************************************************************************
 *
*
 *
*
 *       VDU 12  Clear text Screen               0 parameters
*
 *
*
 *
*

*************************************************************************
 ;
 C759    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C75C    BNE     &C7BD   ;if graphics cursor &C7BD
 C75E    LDA     &D0     ;VDU status byte
 C760    AND     #&08    ;check if software scrolling (text window set)
 C762    BNE     &C767   ;if so C767
 C764    JMP     &CBC1   ;initialise screen display and home cursor
```

```
 C767    LDX     &030B   ;top of text window
 C76A    STX     &0319   ;current text line
 C76D    JSR     &CEAC   ;clear a line

 C770    LDX     &0319   ;current text line
 C773    CPX     &0309   ;bottom margin
 C776    INX             ;X=X+1
 C777    BCC     &C76A   ;if X at compare is less than bottom margin
clear next



*************************************************************************
 *
*
 *
*
 *       VDU 30  Home cursor                     0  parameters
*
 *
*
 *
*

*************************************************************************

 C779    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C77C    BEQ     &C781   ;if text cursor C781
 C77E    JMP     &CFA6   ;home graphic cursor if graphic
 C781    STA     &0323   ;store 0 in last two parameters
 C784    STA     &0322   ;




*************************************************************************
 *
*
 *
*
 *       VDU 31  Position text cursor            2  parameters
*
 *
*
 *
*

*************************************************************************
 ;0322 = X coordinate
 ;0323 = Y coordinate

 C787    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C78A    BNE     &C758   ;exit
 C78C    JSR     &C7A8   ;exchange text column/line with workspace 0328/9
 C78F    CLC             ;clear carry
 C790    LDA     &0322   ;get X coordinate
 C793    ADC     &0308   ;add to text window left
```

```
 C796    STA     &0318   ;store as text column
 C799    LDA     &0323   ;get Y coordinate
 C79C    CLC             ;
 C79D    ADC     &030B   ;add top of text window
 C7A0    STA     &0319   ;current text line
 C7A3    JSR     &CEE8   ;set up screen address
 C7A6    BCC     &C732   ;set cursor position if C=0 (point on screen)
 C7A8    LDX     #&18    ;else point to workspace
 C7AA    LDY     #&28    ;and line/column to restore old values
 C7AC    JMP     &CDDE   ;exchange &300/1+X with &300/1+Y



*************************************************************************
 *
*
 *
*
 *       VDU  13         Carriage  Return        0 parameters
*
 *
*
 *
*

*************************************************************************


 C7AF    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C7B2    BEQ     &C7B7   ;if text C7B7
 C7B4    JMP     &CFAD   ;else set graphics cursor to left hand columm

 C7B7    JSR     &CE6E   ;set text column to left hand column
 C7BA    JMP     &C6AF   ;set up cursor and display address

 C7BD    JSR     &CFA6   ;home graphic cursor



*************************************************************************
 *
*
 *
*
 *       VDU 16 clear graphics screen            0 parameters
*
 *
*
 *
*

*************************************************************************

 C7C0    LDA     &0361   ;pixels per byte
 C7C3    BEQ     &C7F8   ;if 0 current mode has no graphics so exit
```

```
C7C5    LDX     &035A   ;Background graphics colour
C7C8    LDY     &035C   ;background graphics plot mode (GCOL n)
C7CB    JSR     &D0B3   ;set graphics byte mask in &D4/5
C7CE    LDX     #&00    ;graphics window
C7D0    LDY     #&28    ;workspace
C7D2    JSR     &D47C   ;set(300/7+Y) from (300/7+X)
C7D5    SEC             ;set carry
C7D6    LDA     &0306   ;graphics window top lo.
C7D9    SBC     &0302   ;graphics window bottom lo
C7DC    TAY             ;Y=difference
C7DD    INY             ;increment
C7DE    STY     &0330   ;and store in workspace (this is line count)
C7E1    LDX     #&2C    ;
C7E3    LDY     #&28    ;
C7E5    JSR     &D6A6   ;clear line
C7E8    LDA     &032E   ;decrement window height in pixels
C7EB    BNE     &C7F0   ;
C7ED    DEC     &032F   ;
C7F0    DEC     &032E   ;
C7F3    DEC     &0330   ;decrement line count
C7F6    BNE     &C7E1   ;if <>0 then do it again
C7F8    RTS             ;exit
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 17          Define text colour              1 parameter
*
 *       COLOUR
*
 *
*
*************************************************************************
 ;parameter in &0323

C7F9    LDY     #&00    ;Y=0
C7FB    BEQ     &C7FF   ;jump to C7FF
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 18          Define graphics colour          2 parameters
*
 *       GCOL
*
 *
*
```

```
*************************************************************************
 ;parameters in 323,322

 C7FD    LDY     #&02    ;Y=2

 C7FF    LDA     &0323   ;get last parameter
 C802    BPL     &C805   ;if +ve its foreground colour so C805
 C804    INY             ;else Y=Y+1

 C805    AND     &0360   ;number of logical colours less 1
 C808    STA     &DA     ;store it
 C80A    LDA     &0360   ;number of logical colours less 1
 C80D    BEQ     &C82B   ;if none exit
 C80F    AND     #&07    ;else limit to an available colour and clear M
 C811    CLC             ;clear carry
 C812    ADC     &DA     ;Add last parameter to get pointer to table
 C814    TAX             ;pointer into X
 C815    LDA     &C423,X ;get plot options from table
 C818    STA     &0357,Y ; colour Y=0=text fgnd 1= text bkgnd 2=graphics
fg etc
 C81B    CPY     #&02    ;If Y>1
 C81D    BCS     &C82C   ;then its graphics so C82C else
 C81F    LDA     &0357   ;foreground text colour
 C822    EOR     #&FF    ;invert
 C824    STA     &D3     ;text colour byte to be orred or EORed into
memory
 C826    EOR     &0358   ;background text colour
 C829    STA     &D2     ;text colour byte to be orred or EORed into
memory

 C82B    RTS             ;and exit
 ;
 C82C    LDA     &0322   ;get first parameter
 C82F    STA     &0359,Y ;text colour Y=0=foreground 1=background etc.
 C832    RTS             ;exit



 ;
 C833    LDA     #&20    ;
 C835    STA     &0358   ;background text colour
 C838    RTS             ;




*************************************************************************
 *
*
 *
*
 *       VDU 20          Restore default colours         0 parameters
*
 *
*
 *
*
```

```
*****************************************************************************
 ;
 C839    LDX     #&05    ;X=5

 C83B    LDA     #&00    ;A=0
 C83D    STA     &0357,X ;zero all colours
 C840    DEX             ;
 C841    BPL     &C83D   ;until X=&FF
 C843    LDX     &0360   ;number of logical colours less 1
 C846    BEQ     &C833   ;if none its MODE 7 so C833
 C848    LDA     #&FF    ;A=&FF
 C84A    CPX     #&0F    ;if not mode 2 (16 colours)
 C84C    BNE     &C850   ;goto C850

 C84E    LDA     #&3F    ;else A=&3F

 C850    STA     &0357   ;foreground text colour
 C853    STA     &0359   ;foreground graphics colour
 C856    EOR     #&FF    ;invert A
 C858    STA     &D2     ;text colour byte to be orred or EORed into
memory
 C85A    STA     &D3     ;text colour byte to be orred or EORed into
memory
 C85C    STX     &031F   ;set first parameter of 5
 C85F    CPX     #&03    ;if there are 4 colours
 C861    BEQ     &C874   ;goto C874
 C863    BCC     &C885   ;if less there are 2 colours goto C885

                         ;else there are 16 colours
 C865    STX     &0320   ;set second parameter
 C868    JSR     &C892   ;do VDU 19 etc
 C86B    DEC     &0320   ;decrement first parameter
 C86E    DEC     &031F   ;and last parameter
 C871    BPL     &C868   ;
 C873    RTS             ;
 ;
 ********* 4 colour mode
*********************************************

 C874    LDX     #&07    ;X=7
 C876    STX     &0320   ;set first parameter
 C879    JSR     &C892   ;and do VDU 19
 C87C    LSR     &0320   ;
 C87F    DEC     &031F   ;
 C882    BPL     &C879   ;
 C884    RTS             ;exit

 ;********* 2 colour mode
*********************************************

 C885    LDX     #&07    ;X=7
 C887    JSR     &C88F   ;execute VDU 19
 C88A    LDX     #&00    ;X=0
 C88C    STX     &031F   ;store it as
 C88F    STX     &0320   ;both parameters
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 19   define logical colours                 5 parameters
*
 *
*
 *
*

*************************************************************************
 ;&31F=first parameter logical colour
 ;&320=second physical colour

 C892    PHP             ;push processor flags
 C893    SEI             ;disable interrupts
 C894    LDA     &031F   ;get first parameter and
 C897    AND     &0360   ;number of logical colours less 1
 C89A    TAX             ;toi make legal  X=A
 C89B    LDA     &0320   ;A=second parameter
 C89E    AND     #&0F    ;make legal
 C8A0    STA     &036F,X ;colour pallette
 C8A3    TAY             ;Y=A
 C8A4    LDA     &0360   ;number of logical colours less 1
 C8A7    STA     &FA     ;store it
 C8A9    CMP     #&03    ;is it 4 colour mode??
 C8AB    PHP             ;save flags
 C8AC    TXA             ;A=X
 C8AD    ROR             ;rotate A into &FA
 C8AE    ROR     &FA     ;
 C8B0    BCS     &C8AD   ;
 C8B2    ASL     &FA     ;
 C8B4    TYA             ;A=Y
 C8B5    ORA     &FA     ;
 C8B7    TAX             ;
 C8B8    LDY     #&00    ;Y=0
 C8BA    PLP             ;check flags
 C8BB    PHP             ;
 C8BC    BNE     &C8CC   ;if A<>3 earlier C8CC
 C8BE    AND     #&60    ;else A=&60 to test bits 5 and 6
 C8C0    BEQ     &C8CB   ;if not set C8CB
 C8C2    CMP     #&60    ;else if both set
 C8C4    BEQ     &C8CB   ;C8CB
 C8C6    TXA             ;A=X
 C8C7    EOR     #&60    ;invert
 C8C9    BNE     &C8CC   ;and if not 0 C8CC

 C8CB    TXA             ;X=A
 C8CC    JSR     &EA11   ;call Osbyte 155 pass data to pallette register
 C8CF    TYA             ;
 C8D0    SEC             ;
 C8D1    ADC     &0360   ;number of logical colours less 1
 C8D4    TAY             ;
 C8D5    TXA             ;
 C8D6    ADC     #&10    ;
 C8D8    TAX             ;
 C8D9    CPY     #&10    ;if Y<16 do it again
```

```
 C8DB    BCC     &C8BA   ;
 C8DD    PLP             ;pull flags twice
 C8DE    PLP             ;
 C8DF    RTS             ;and exit




*************************************************************************
 *
*
 *
*
 *       OSWORD 12    WRITE PALLETTE
*
 *
*
 *
*

*************************************************************************
 ;on entry X=&F0:Y=&F1:YX points to parameter block
 ;byte 0 = logical colour;  byte 1 physical colour; bytes 2-4=0

 C8E0    PHP             ;push flags
 C8E1    AND     &0360   ;and with number of logical colours less 1
 C8E4    TAX             ;X=A
 C8E5    INY             ;Y=Y+1
 C8E6    LDA     (&F0),Y ;get phsical colour
 C8E8    JMP     &C89E   ;do VDU19 with parameters in X and A




*************************************************************************
 *
*
 *
*
 *       VDU     22              Select Mode   1 parameter
*
 *
*
 *
*

*************************************************************************
 ;parameter in &323




 C8EB    LDA     &0323   ;get parameter
 C8EE    JMP     &CB33   ;goto CB33
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 23 Define characters                9 parameters
*
 *
*
 *
*

*************************************************************************
 ;parameters are:-
 ;31B character to define
 ;31C to 323 definition

 C8F1    LDA     &031B   ;get character to define
 C8F4    CMP     #&20    ;is it ' '
 C8F6    BCC     &C93F   ;if less then it is an instruction to set CRT
                         ;controller goto C93F
 C8F8    PHA             ;else save parameter
 C8F9    LSR             ;A=A/32
 C8FA    LSR             ;
 C8FB    LSR             ;
 C8FC    LSR             ;
 C8FD    LSR             ;
 C8FE    TAX             ;X=A
 C8FF    LDA     &C40D,X ;get font flag mask from table (A=&80/2^X)
 C902    BIT     &0367   ;font flag
 C905    BNE     &C927   ;and if A<>0 C927 storage area is established
already
 C907    ORA     &0367   ;or with font flag to set bit found to be 0
 C90A    STA     &0367   ;font flag
 C90D    TXA             ;get back A
 C90E    AND     #&03    ;And 3 to clear all but bits 0 and 1
 C910    CLC             ;clear carry
 C911    ADC     #&BF    ;add &BF (A=&C0,&C1,&C2) to select a character
page
 C913    STA     &DF     ;store it
 C915    LDA     &0367,X ;get font location byte (normally &0C)
 C918    STA     &DD     ;store it
 C91A    LDY     #&00    ;Y=0 so (&DE) holds (&C000 -&C2FF)
 C91C    STY     &DC     ;
 C91E    STY     &DE     ;

 C920    LDA     (&DE),Y ;transfer page to storage area
 C922    STA     (&DC),Y ;
 C924    DEY             ;
 C925    BNE     &C920   ;

 C927    PLA             ;get back A
 C928    JSR     &D03E   ;set up character definition pointers

 C92B    LDY     #&07    ;Y=7
```

```
 C92D    LDA     &031C,Y ;transfer definition parameters
 C930    STA     (&DE),Y ;to RAM definition
 C932    DEY             ;
 C933    BPL     &C92D   ;

 C935    RTS             ;and exit

 ;
 C936    PLA             ;Pull A
 C937    RTS             ;and exit
 ;



 ************ VDU EXTENSION
*****************************************

 C938    LDA     &031F   ;A=fifth VDU parameter
 C93B    CLC             ;clear carry
 C93C    JMP     (&0226) ;jump via VDUV vector


 ********** set CRT controller
*****************************************

 C93F    CMP     #&01    ;does A=1
 C941    BCC     &C958   ;if less (0) then set CRT register

 C943    BNE     &C93C   ;if not 1 jump to VDUV

 C945    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
 C948    BNE     &C937   ;if graphics exit
 C94A    LDA     #&20    ;else A=&20
 C94C    LDY     &031C   ;Y=second VDU parameter
 C94F    BEQ     &C954   ;if 0 C954
 C951    LDA     &035F   ;last setting of CRT controller register

 C954    LDY     #&0A    ;Y=10
 C956    BNE     &C985   ;jump to C985

 C958    LDA     &031D   ;get third
 C95B    LDY     &031C   ;and second parameter
 C95E    CPY     #&07    ;is Y=7
 C960    BCC     &C985   ;if less C985
 C962    BNE     &C967   ;else if >7 C967
 C964    ADC     &0290   ;else ADD screen vertical display adjustment

 C967    CPY     #&08    ;If Y<>8
 C969    BNE     &C972   ;C972
 C96B    ORA     #&00    ;if bit 7 set
 C96D    BMI     &C972   ;C972
 C96F    EOR     &0291   ;else EOR with interlace toggle

 C972    CPY     #&0A    ;Y=10??
 C974    BNE     &C985   ;if not C985
 C976    STA     &035F   ;last setting of CRT controller register
 C979    TAY             ;Y=A
 C97A    LDA     &D0     ;VDU status byte
 C97C    AND     #&20    ;check bit 5 printing at graphics cursor??
 C97E    PHP             ;push flags
```

```
 C97F    TYA             ;Y=A
 C980    LDY     #&0A    ;Y=10
 C982    PLP             ;pull flags
 C983    BNE     &C98B   ;if graphics in use then C98B

 C985    STY     &FE00   ;else set CRTC address register
 C988    STA     &FE01   ;and poke new value to register Y
 C98B    RTS             ;exit




*************************************************************************
 *
*
 *
*
 *       VDU 25          PLOT                    5 parameters
*
 *
*
 *
*

*************************************************************************
 ;
 C98C    LDX     &0361   ;pixels per byte
 C98F    BEQ     &C938   ;if no graphics available go via VDU Extension
 C991    JMP     &D060   ;else enter Plot routine at D060


 ********** adjust screen RAM addresses
**********************************

 C994    LDX     &0350   ;window area start address lo
 C997    LDA     &0351   ;window area start address hi
 C99A    JSR     &CCF8   ;subtract bytes per character row from this
 C99D    BCS     &C9B3   ;if no wraparound needed C9B3

 C99F    ADC     &0354   ;screen RAM size hi byte to wrap around
 C9A2    BCC     &C9B3   ;

 C9A4    LDX     &0350   ;window area start address lo
 C9A7    LDA     &0351   ;window area start address hi
 C9AA    JSR     &CAD4   ;add bytes per char. row
 C9AD    BPL     &C9B3   ;

 C9AF    SEC             ;wrap around i other direction
 C9B0    SBC     &0354   ;screen RAM size hi byte
 C9B3    STA     &0351   ;window area start address hi
 C9B6    STX     &0350   ;window area start address lo
 C9B9    LDY     #&0C    ;Y=12
 C9BB    BNE     &CA0E   ;jump to CA0E




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

```
 *
*
 *
*
 *       VDU 26  set default windows             0 parameters
*
 *
*
 *
*

*********************************************************************


 C9BD    LDA     #&00    ;A=0
 C9BF    LDX     #&2C    ;X=&2C

 C9C1    STA     &0300,X ;clear all windows
 C9C4    DEX             ;
 C9C5    BPL     &C9C1   ;until X=&FF

 C9C7    LDX     &0355   ;screen mode
 C9CA    LDY     C3EF,X  ;text window right hand margin maximum
 C9CD    STY     &030A   ;text window right
 C9D0    JSR     &CA88   ;calculate number of bytes in a line
 C9D3    LDY     &C3E7,X ;text window bottom margin maximum
 C9D6    STY     &0309   ;bottom margin
 C9D9    LDY     #&03    ;Y=3
 C9DB    STY     &0323   ;set as last parameter
 C9DE    INY             ;increment Y
 C9DF    STY     &0321   ;set parameters
 C9E2    DEC     &0322   ;
 C9E5    DEC     &0320   ;
 C9E8    JSR     &CA39   ;and do VDU 24
 C9EB    LDA     #&F7    ;
 C9ED    JSR     &C5A8   ;clear bit 3 of &D0
 C9F0    LDX     &0350   ;window area start address lo
 C9F3    LDA     &0351   ;window area start address hi
 C9F6    STX     &034A   ;text cursor 6845 address
 C9F9    STA     &034B   ;text cursor 6845 address
 C9FC    BPL     &CA02   ;set cursor position
 C9FE    SEC             ;
 C9FF    SBC     &0354   ;screen RAM size hi byte


 ********** set cursor position
***********************************


 CA02    STX     &D8     ;set &D8/9 from X/A
 CA04    STA     &D9     ;
 CA06    LDX     &034A   ;text cursor 6845 address
 CA09    LDA     &034B   ;text cursor 6845 address
 CA0C    LDY     #&0E    ;Y=15
 CA0E    PHA             ;Push A
 CA0F    LDA     &0355   ;screen mode
 CA12    CMP     #&07    ;is it mode 7?
 CA14    PLA             ;get back A
 CA15    BCS     &CA27   ;if mode 7 selected CA27
 CA17    STX     &DA     ;else store X
```

```
CA19    LSR             ;divide X/A by 8
CA1A    ROR     &DA     ;
CA1C    LSR             ;
CA1D    ROR     &DA     ;
CA1F    LSR             ;
CA20    ROR     &DA     ;
CA22    LDX     &DA     ;
CA24    JMP     &CA2B   ;goto CA2B

CA27    SBC     #&74    ;mode 7 subtract &74
CA29    EOR     #&20    ;EOR with &20
CA2B    STY     &FE00   ;write to CRTC address file register
CA2E    STA     &FE01   ;and to relevant address (register 14)
CA31    INY             ;Increment Y
CA32    STY     &FE00   ;write to CRTC address file register
CA35    STX     &FE01   ;and to relevant address (register 15)
CA38    RTS             ;and RETURN
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 24 Define graphics window           8 parameters
*
 *
*
 *
*

*************************************************************************
 ;&31C/D Left margin
 ;&31E/F Bottom margin
 ;&320/1 Right margin
 ;&322/3 Top margin

 CA39    JSR     &CA81   ;exchange 310/3 with 328/3
 CA3C    LDX     #&1C    ;
 CA3E    LDY     #&2C    ;
 CA40    JSR     &D411   ;calculate width=right - left
                         ;          height = top-bottom
 CA43    ORA     &032D   ;
 CA46    BMI     &CA81   ;exchange 310/3 with 328/3 and exit
 CA48    LDX     #&20    ;X=&20
 CA4A    JSR     &D149   ;scale pointers to mode
 CA4D    LDX     #&1C    ;X=&1C
 CA4F    JSR     &D149   ;scale pointers to mode
 CA52    LDA     &031F   ;check for negative margins
 CA55    ORA     &031D   ;
 CA58    BMI     &CA81   ;if found exchange 310/3 with 328/3 and exit
 CA5A    LDA     &0323   ;
 CA5D    BNE     &CA81   ;exchange 310/3 with 328/3 and exit
 CA5F    LDX     &0355   ;screen mode
 CA62    LDA     &0321   ;right margin hi
 CA65    STA     &DA     ;store it
 CA67    LDA     &0320   ;right margin lo
 CA6A    LSR     &DA     ;/2
 CA6C    ROR             ;A=A/2
 CA6D    LSR     &DA     ;/2
 CA6F    BNE     &CA81   ;exchange 310/3 with 328/3
 CA71    ROR             ;A=A/2
 CA72    LSR             ;A=A/2
 CA73    CMP     C3EF,X  ;text window right hand margin maximum
 CA76    BEQ     &CA7A   ;if equal CA7A
 CA78    BPL     &CA81   ;exchange 310/3 with 328/3

 CA7A    LDY     #&00    ;Y=0
 CA7C    LDX     #&1C    ;X=&1C
 CA7E    JSR     &D47C   ;set(300/7+Y) from (300/7+X)


 ***************** exchange 310/3 with 328/3
*****************************

 CA81    LDX     #&10    ;X=10
 CA83    LDY     #&28    ;Y=&28
 CA85    JMP     &CDE6   ;exchange 300/3+Y and 300/3+X
```

```
CA88    INY             ;Y=Y+1
CA89    TYA             ;A=Y
CA8A    LDY     #&00    ;Y=0
CA8C    STY     &034D   ;text window width hi (bytes)
CA8F    STA     &034C   ;text window width lo (bytes)
CA92    LDA     &034F   ;bytes per character
CA95    LSR             ;/2
CA96    BEQ     &CAA1   ;if 0 exit
CA98    ASL     &034C   ;text window width lo (bytes)
CA9B    ROL     &034D   ;text window width hi (bytes)
CA9E    LSR             ;/2
CA9F    BCC     &CA98   ;
CAA1    RTS             ;
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 29  Set graphics origin                     4 parameters
*
 *
*
 *
*
```

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

```
 ;
CAA2    LDX     #&20    ;
CAA4    LDY     #&0C    ;
CAA6    JSR     &D48A   ;(&300/3+Y)=(&300/3+X)
CAA9    JMP     &D1B8   ;set up external coordinates for graphics
```

```
*************************************************************************
 *
*
 *
*
 *       VDU 32  (&7F)   Delete                          0 parameters
*
 *
*
 *
*
```

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

```
CAAC    JSR     &C5C5   ;cursor left
CAAF    JSR     &C588   ;A=0 if text cursor A=&20 if graphics cursor
CAB2    BNE     &CAC7   ;if graphics then CAC7
```

```
 CAB4    LDX     &0360   ;number of logical colours less 1
 CAB7    BEQ     &CAC2   ;if mode 7 CAC2
 CAB9    STA     &DE     ;else store A (always 0)
 CABB    LDA     #&C0    ;A=&C0
 CABD    STA     &DF     ;store in &DF (&DE) now points to C300 SPACE
pattern
 CABF    JMP     &CFBF   ;display a space

 CAC2    LDA     #&20    ;A=&20
 CAC4    JMP     &CFDC   ;and return to display a space

 CAC7    LDA     #&7F    ;for graphics cursor
 CAC9    JSR     &D03E   ;set up character definition pointers
 CACC    LDX     &035A   ;Background graphics colour
 CACF    LDY     #&00    ;Y=0
 CAD1    JMP     &CF63   ;invert pattern data (to background colour)


 ***** Add number of bytes in a line to X/A
*******************************

 CAD4    PHA             ;store A
 CAD5    TXA             ;A=X
 CAD6    CLC             ;clear carry
 CAD7    ADC     &0352   ;bytes per character row
 CADA    TAX             ;X=A
 CADB    PLA             ;get back A
 CADC    ADC     &0353   ;bytes per character row
 CADF    RTS             ;and return
 ;
 ********* control scrolling in paged mode
*******************************

 CAE0    JSR     &CB14   ;zero paged mode line counter
 CAE3    JSR     &E9D9   ;osbyte 118 check keyboard status; set LEDs
 CAE6    BCC     &CAEA   ;if carry clear CAEA
 CAE8    BMI     &CAE0   ;if M set CAE0 do it again

 CAEA    LDA     &D0     ;VDU status byte
 CAEC    EOR     #&04    ;invert bit 2 paged scrolling
 CAEE    AND     #&46    ;and if 2 cursors, paged mode off, or scrolling
 CAF0    BNE     &CB1C   ;barred then CB1C to exit

 CAF2    LDA     &0269   ;paged mode counter
 CAF5    BMI     &CB19   ;if negative then exit via CB19

 CAF7    LDA     &0319   ;current text line
 CAFA    CMP     &0309   ;bottom margin
 CAFD    BCC     &CB19   ;increment line counter and exit

 CAFF    LSR             ;A=A/4
 CB00    LSR             ;
 CB01    SEC             ;set carry
 CB02    ADC     &0269   ;paged mode counter
 CB05    ADC     &030B   ;top of text window
 CB08    CMP     &0309   ;bottom margin
 CB0B    BCC     &CB19   ;increment line counter and exit

 CB0D    CLC             ;clear carry
 CB0E    JSR     &E9D9   ;osbyte 118 check keyboard status; set LEDs
```

```
CB11    SEC             ;set carry
CB12    BPL     &CB0E   ;if +ve result then loop till shift pressed


*************** zero paged mode  counter
******************************

CB14    LDA     #&FF    ;
CB16    STA     &0269   ;paged mode counter
CB19    INC     &0269   ;paged mode counter
CB1C    RTS             ;
;

*********part of intitialisation routines ******************************

CB1D    PHA             ;save A
CB1E    LDX     #&7F    ;X=&7F
CB20    LDA     #&00    ;A=0
CB22    STA     &D0     ;VDU status byte to set default conditions

CB24    STA     &02FF,X ;zero 300,37E
CB27    DEX             ;with this loop
CB28    BNE     &CB24   ;

CB2A    JSR     &CD07   ;implode character definitions
CB2D    PLA             ;get back A
CB2E    LDX     #&7F    ;X=&7F
CB30    STX     &0366   ;mode 7 write cursor character
CB33    BIT     &028E   ;available RAM pages
CB36    BMI     &CB3A   ;if 32k CB3A

CB38    ORA     #&04    ;ensure only modes 4-7 are available

CB3A    AND     #&07    ;X=A and 7 ensure legal mode
CB3C    TAX             ;X=mode
CB3D    STX     &0355   ;set screen mode flag
CB40    LDA     &C414,X ;no. of colours -1 in mode table
CB43    STA     &0360   ;number of logical colours less 1
CB46    LDA     &C3FF,X ;number of bytes /character for each mode
CB49    STA     &034F   ;bytes per character
CB4C    LDA     &C43A,X ;display mode pixels/byte table
CB4F    STA     &0361   ;pixels per byte
CB52    BNE     &CB56   ;if <> 0 CB56
CB54    LDA     #&07    ;else A=7

CB56    ASL             ;A=A*2
CB57    TAY             ;Y=A
CB58    LDA     &C406,Y ;mask table
CB5B    STA     &0363   ;colour mask left
CB5E    ASL             ;A=A*2
CB5F    BPL     &CB5E   ;If still +ve CB5E
CB61    STA     &0362   ;colour mask right
CB64    LDY     &C440,X ;screen display memory index table
CB67    STY     &0356   ;memory map type
CB6A    LDA     &C44F,Y ;VDU section control
CB6D    JSR     &E9F8   ;set hardware scrolling to VIA
CB70    LDA     &C44B,Y ;VDU section control
CB73    JSR     &E9F8   ;set hardware scrolling to VIA
CB76    LDA     &C459,Y ;Screen RAM size hi byte table
CB79    STA     &0354   ;screen RAM size hi byte
```

```
CB7C    LDA     &C45E,Y ;screen ram address hi byte
CB7F    STA     &034E   ;hi byte of screen RAM address
CB82    TYA             ;Y=A
CB83    ADC     #&02    ;Add 2
CB85    EOR     #&07    ;
CB87    LSR             ;/2
CB88    TAX             ;X=A
CB89    LDA     &C466,X ;row multiplication table pointer
CB8C    STA     &E0     ;store it
CB8E    LDA     #&C3    ;A=&C3
CB90    STA     &E1     ;store it (&E0) now points to C3B5 or C375
CB92    LDA     &C463,X ;get nuber of bytes per row from table
CB95    STA     &0352   ;store as bytes per character row
CB98    STX     &0353   ;bytes per character row
CB9B    LDA     #&43    ;A=&43
CB9D    JSR     &C5A8   ;A=A and &D0:&D0=A
CBA0    LDX     &0355   ;screen mode
CBA3    LDA     &C3F7,X ;get video ULA control setting
CBA6    JSR     &EA00   ;set video ULA using osbyte 154
CBA9    PHP             ;push flags
CBAA    SEI             ;set interrupts
CBAB    LDX     &C469,Y ;get cursor end register data from table
CBAE    LDY     #&0B    ;Y=11

CBB0    LDA     &C46E,X ;get end of 6845 registers 0-11 table
CBB3    JSR     &C95E   ;set register Y
CBB6    DEX             ;reduce pointers
CBB7    DEY             ;
CBB8    BPL     &CBB0   ;and if still >0 do it again

CBBA    PLP             ;pull flags
CBBB    JSR     &C839   ;set default colours
CBBE    JSR     &C9BD   ;set default windows

CBC1    LDX     #&00    ;X=0
CBC3    LDA     &034E   ;hi byte of screen RAM address
CBC6    STX     &0350   ;window area start address lo
CBC9    STA     &0351   ;window area start address hi
CBCC    JSR     &C9F6   ;use X and Y to set new cursor address
CBCF    LDY     #&0C    ;Y=12
CBD1    JSR     &CA2B   ;set registers 12 and 13 in CRTC
CBD4    LDA     &0358   ;background text colour
CBD7    LDX     &0356   ;memory map type
CBDA    LDY     &C454,X ;get section control number
CBDD    STY     &035D   ;set it in jump vector lo
CBE0    LDY     #&CC    ;Y=&CC
CBE2    STY     &035E   ;upper byte of link address
CBE5    LDX     #&00    ;X=0
CBE7    STX     &0269   ;paged mode counter
CBEA    STX     &0318   ;text column
CBED    STX     &0319   ;current text line
CBF0    JMP     (&035D) ;jump vector set up previously
```

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

```
 *
*
 *       OSWORD 10       Read character definition
*
 *
*
 *
*

*********************************************************************
 ;&EF=A:&F0=X:&F1=Y, on entry YX contains number of byte to be read
 ;(&DE) points to address
 ;on exit byte YX+1 to YX+8 contain definition

 CBF3    JSR     &D03E   ;set up character definition pointers
 CBF6    LDY     #&00    ;Y=0
 CBF8    LDA     (&DE),Y ;get first byte
 CBFA    INY             ;Y=Y+1
 CBFB    STA     (&F0),Y ;store it in YX
 CBFD    CPY     #&08    ;until Y=8
 CBFF    BNE     &CBF8   ;
 CC01    RTS             ;then exit
 ;




*********************************************************************
 *
*
 *       MAIN SCREEN CLEARANCE ROUTINE
*
 *
*

*********************************************************************
 ;on entry A contains background colour which is set in every byte
 ;of the screen


 ************************ Mode 0,1,2 entry point
************************

 CC02    STA     &3000,X ;
 CC05    STA     &3100,X ;
 CC08    STA     &3200,X ;
 CC0B    STA     &3300,X ;
 CC0E    STA     &3400,X ;
 CC11    STA     &3500,X ;
 CC14    STA     &3600,X ;
 CC17    STA     &3700,X ;
 CC1A    STA     &3800,X ;
 CC1D    STA     &3900,X ;
 CC20    STA     &3A00,X ;
 CC23    STA     &3B00,X ;
 CC26    STA     &3C00,X ;
 CC29    STA     &3D00,X ;
 CC2C    STA     &3E00,X ;
 CC2F    STA     &3F00,X ;
```

```
 ************************ Mode 3 entry point
*****************************

 CC32    STA     &4000,X ;
 CC35    STA     &4100,X ;
 CC38    STA     &4200,X ;
 CC3B    STA     &4300,X ;
 CC3E    STA     &4400,X ;
 CC41    STA     &4500,X ;
 CC44    STA     &4600,X ;
 CC47    STA     &4700,X ;
 CC4A    STA     &4800,X ;
 CC4D    STA     &4900,X ;
 CC50    STA     &4A00,X ;
 CC53    STA     &4B00,X ;
 CC56    STA     &4C00,X ;
 CC59    STA     &4D00,X ;
 CC5C    STA     &4E00,X ;
 CC5F    STA     &4F00,X ;
 CC62    STA     &5000,X ;
 CC65    STA     &5100,X ;
 CC68    STA     &5200,X ;
 CC6B    STA     &5300,X ;
 CC6E    STA     &5400,X ;
 CC71    STA     &5500,X ;
 CC74    STA     &5600,X ;
 CC77    STA     &5700,X ;


 ************************ Mode 4,5 entry point
*****************************

 CC7A    STA     &5800,X ;
 CC7D    STA     &5900,X ;
 CC80    STA     &5A00,X ;
 CC83    STA     &5B00,X ;
 CC86    STA     &5C00,X ;
 CC89    STA     &5D00,X ;
 CC8C    STA     &5E00,X ;
 CC8F    STA     &5F00,X ;


 ************************ Mode 6 entry point
*****************************

 CC92    STA     &6000,X ;
 CC95    STA     &6100,X ;
 CC98    STA     &6200,X ;
 CC9B    STA     &6300,X ;
 CC9E    STA     &6400,X ;
 CCA1    STA     &6500,X ;
 CCA4    STA     &6600,X ;
 CCA7    STA     &6700,X ;
 CCAA    STA     &6800,X ;
 CCAD    STA     &6900,X ;
 CCB0    STA     &6A00,X ;
 CCB3    STA     &6B00,X ;
```

```
 CCB6    STA     &6C00,X ;
 CCB9    STA     &6D00,X ;
 CCBC    STA     &6E00,X ;
 CCBF    STA     &6F00,X ;
 CCC2    STA     &7000,X ;
 CCC5    STA     &7100,X ;
 CCC8    STA     &7200,X ;
 CCCB    STA     &7300,X ;
 CCCE    STA     &7400,X ;
 CCD1    STA     &7500,X ;
 CCD4    STA     &7600,X ;
 CCD7    STA     &7700,X ;
 CCDA    STA     &7800,X ;
 CCDD    STA     &7900,X ;
 CCE0    STA     &7A00,X ;
 CCE3    STA     &7B00,X ;


 ************************ Mode 7 entry point
*******************************

 CCE6    STA     &7C00,X ;
 CCE9    STA     &7D00,X ;
 CCEC    STA     &7E00,X ;
 CCEF    STA     &7F00,X ;
 CCF2    INX             ;
 CCF3    BEQ     &CD65   ;exit


 ****************** execute required function
****************************

 CCF5    JMP     (&035D) ;jump vector set up previously


 ********* subtract bytes per line from X/A
*******************************

 CCF8    PHA             ;Push A
 CCF9    TXA             ;A=X
 CCFA    SEC             ;set carry for subtraction
 CCFB    SBC     &0352   ;bytes per character row
 CCFE    TAX             ;restore X
 CCFF    PLA             ;and A
 CD00    SBC     &0353   ;bytes per character row
 CD03    CMP     &034E   ;hi byte of screen RAM address
 CD06    RTS             ;return




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

```
 *       OSBYTE 20              Explode characters
*
 *
*
 *
*

*********************************************************************

 ;
 CD07    LDA     #&0F    ;A=15
 CD09    STA     &0367   ;font flag indicating that page &0C,&C0-&C2 are
                         ;used for user defined characters
 CD0C    LDA     #&0C    ;A=&0C
 CD0E    LDY     #&06    ;set loop counter

 CD10    STA     &0368,Y ;set all font location bytes
 CD13    DEY             ;to page &0C to indicate only page available
 CD14    BPL     &CD10   ;for user character definitions

 CD16    CPX     #&07    ;is X= 7 or greater
 CD18    BCC     &CD1C   ;if not CD1C
 CD1A    LDX     #&06    ;else X=6
 CD1C    STX     &0246   ;character definition explosion switch
 CD1F    LDA     &0243   ;A=primary OSHWM
 CD22    LDX     #&00    ;X=0

 CD24    CPX     &0246   ;character definition explosion switch
 CD27    BCS     &CD34   ;
 CD29    LDY     &C4BA,X ;get soft character  RAM allocation
 CD2C    STA     &0368,Y ;font location bytes
 CD2F    ADC     #&01    ;Add 1
 CD31    INX             ;X=X+1
 CD32    BNE     &CD24   ;if X<>0 then CD24

 CD34    STA     &0244   ;current value of page (OSHWM)
 CD37    TAY             ;Y=A
 CD38    BEQ     &CD06   ;return via CD06 (ERROR?)

 CD3A    LDX     #&11    ;X=&11
 CD3C    JMP     &F168   ;issue paged ROM service call &11
                         ;font implosion/explosion warning

 ********:move text cursor to next line
*********************************

 CD3F    LDA     #&02    ;A=2 to check if scrolling disabled
 CD41    BIT     &D0     ;VDU status byte
 CD43    BNE     &CD47   ;if scrolling is barred CD47
 CD45    BVC     &CD79   ;if cursor editing mode disabled RETURN

 CD47    LDA     &0309   ;bottom margin
 CD4A    BCC     &CD4F   ;if carry clear on entry CD4F
 CD4C    LDA     &030B   ;else if carry set get top of text window
 CD4F    BVS     &CD59   ;and if cursor editing enabled CD59
 CD51    STA     &0319   ;get current text line
 CD54    PLA             ;pull return link from stack
 CD55    PLA             ;
 CD56    JMP     &C6AF   ;set up cursor and display address
```

```
 CD59    PHP             ;push flags
 CD5A    CMP     &0365   ;Y coordinate of text input cursor
 CD5D    BEQ     &CD78   ;if A=line count of text input cursor CD78 to
exit
 CD5F    PLP             ;get back flags
 CD60    BCC     &CD66   ;
 CD62    DEC     &0365   ;Y coordinate of text input cursor

 CD65    RTS             ;exit
 ;
 CD66    INC     &0365   ;Y coordinate of text input cursor
 CD69    RTS             ;exit



 ********************** set up write cursor
******************************

 CD6A    PHP             ;save flags
 CD6B    PHA             ;save A
 CD6C    LDY     &034F   ;bytes per character
 CD6F    DEY             ;Y=Y-1
 CD70    BNE     &CD8F   ;if Y=0 Mode 7 is in use

 CD72    LDA     &0338   ;so get mode 7 write character cursor character
&7F
 CD75    STA     (&D8),Y ;store it at top scan line of current character
 CD77    PLA             ;pull A
 CD78    PLP             ;pull flags
 CD79    RTS             ;and exit
 ;
 CD7A    PHP             ;push flags
 CD7B    PHA             ;push A
 CD7C    LDY     &034F   ;bytes per character
 CD7F    DEY             ;
 CD80    BNE     &CD8F   ;if not mode 7
 CD82    LDA     (&D8),Y ;get cursor from top scan line
 CD84    STA     &0338   ;store it
 CD87    LDA     &0366   ;mode 7 write cursor character
 CD8A    STA     (&D8),Y ;store it at scan line
 CD8C    JMP     &CD77   ;and exit

 CD8F    LDA     #&FF    ;A=&FF =cursor
 CD91    CPY     #&1F    ;except in mode 2 (Y=&1F)
 CD93    BNE     &CD97   ;if not CD97
 CD95    LDA     #&3F    ;load cursor byte mask


 ********** produce white block write cursor
******************************

 CD97    STA     &DA     ;store it
 CD99    LDA     (&D8),Y ;get scan line byte
 CD9B    EOR     &DA     ;invert it
 CD9D    STA     (&D8),Y ;store it on scan line
 CD9F    DEY             ;decrement scan line counter
 CDA0    BPL     &CD99   ;do it again
 CDA2    BMI     &CD77   ;then jump to &CD77
```

```
 CDA4    JSR     &CE5B   ;exchange line and column cursors with workspace
copies
 CDA7    LDA     &0309   ;bottom margin
 CDAA    STA     &0319   ;current text line
 CDAD    JSR     &CF06   ;set up display address
 CDB0    JSR     &CCF8   ;subtract bytes per character row from this
 CDB3    BCS     &CDB8   ;wraparound if necessary
 CDB5    ADC     &0354   ;screen RAM size hi byte
 CDB8    STA     &DB     ;store A
 CDBA    STX     &DA     ;X
 CDBC    STA     &DC     ;A again
 CDBE    BCS     &CDC6   ;if C set there was no wraparound so CDC6
 CDC0    JSR     &CE73   ;copy line to new position
                         ;using (&DA) for read
                         ;and (&D8) for write
 CDC3    JMP     &CDCE   ;

 CDC6    JSR     &CCF8   ;subtract bytes per character row from X/A
 CDC9    BCC     &CDC0   ;if a result is outside screen RAM CDC0
 CDCB    JSR     &CE38   ;perform a copy

 CDCE    LDA     &DC     ;set write pointer from read pointer
 CDD0    LDX     &DA     ;
 CDD2    STA     &D9     ;
 CDD4    STX     &D8     ;
 CDD6    DEC     &DE     ;decrement window height
 CDD8    BNE     &CDB0   ;and if not zero CDB0
 CDDA    LDX     #&28    ;point to workspace
 CDDC    LDY     #&18    ;point to text column/line
 CDDE    LDA     #&02    ;number of bytes to swap
 CDE0    BNE     &CDE8   ;exchange (328/9)+Y with (318/9)+X
 CDE2    LDX     #&24    ;point to graphics cursor
 CDE4    LDY     #&14    ;point to last graphics cursor
                         ;A=4 to swap X and Y coordinates


 *************** exchange 300/3+Y with 300/3+X
**************************

 CDE6    LDA     #&04    ;A =4


 ************** exchange (300/300+A)+Y with (300/300+A)+X
*****************

 CDE8    STA     &DA     ;store it as loop counter

 CDEA    LDA     &0300,X ;get byte
 CDED    PHA             ;store it
 CDEE    LDA     &0300,Y ;get byte pointed to by Y
 CDF1    STA     &0300,X ;put it in 300+X
 CDF4    PLA             ;get back A
 CDF5    STA     &0300,Y ;put it in 300+Y
 CDF8    INX             ;increment pointers
 CDF9    INY             ;
 CDFA    DEC     &DA     ;decrement loop counter
 CDFC    BNE     &CDEA   ;and if not 0 do it again
 CDFE    RTS             ;and exit
```

```
 ******** execute upward scroll
****************************************
 ;
 CDFF    JSR     &CE5B   ;exchange line and column cursors with workspace
copies
 CE02    LDY     &030B   ;top of text window
 CE05    STY     &0319   ;current text line
 CE08    JSR     &CF06   ;set up display address
 CE0B    JSR     &CAD4   ;add bytes per char. row
 CE0E    BPL     &CE14   ;
 CE10    SEC             ;
 CE11    SBC     &0354   ;screen RAM size hi byte

 CE14    STA     &DB     ;(&DA)=X/A
 CE16    STX     &DA     ;
 CE18    STA     &DC     ;&DC=A
 CE1A    BCC     &CE22   ;
 CE1C    JSR     &CE73   ;copy line to new position
                         ;using (&DA) for read
                         ;and (&D8) for write
 CE1F    JMP     &CE2A   ;exit


 CE22    JSR     &CAD4   ;add bytes per char. row
 CE25    BMI     &CE1C   ;if outside screen RAM CE1C
 CE27    JSR     &CE38   ;perform a copy
 CE2A    LDA     &DC     ;
 CE2C    LDX     &DA     ;
 CE2E    STA     &D9     ;
 CE30    STX     &D8     ;
 CE32    DEC     &DE     ;decrement window height
 CE34    BNE     &CE0B   ;CE0B if not 0
 CE36    BEQ     &CDDA   ;exchange text column/linelse CDDA


 *********** copy routines
***********************************************

 CE38    LDX     &034D   ;text window width hi (bytes)
 CE3B    BEQ     &CE4D   ;if no more than 256 bytes to copy X=0 so CE4D

 CE3D    LDY     #&00    ;Y=0 to set loop counter

 CE3F    LDA     (&DA),Y ;copy 256 bytes
 CE41    STA     (&D8),Y ;
 CE43    INY             ;
 CE44    BNE     &CE3F   ;Till Y=0 again
 CE46    INC     &D9     ;increment hi bytes
 CE48    INC     &DB     ;
 CE4A    DEX             ;decrement window width
 CE4B    BNE     &CE3F   ;if not 0 go back and do loop again

 CE4D    LDY     &034C   ;text window width lo (bytes)
 CE50    BEQ     &CE5A   ;if Y=0 CE5A

 CE52    DEY             ;else Y=Y-1
 CE53    LDA     (&DA),Y ;copy Y bytes
 CE55    STA     (&D8),Y ;
 CE57    TYA             ;A=Y
 CE58    BNE     &CE52   ;if not 0 CE52
```

```
CE5A    RTS             ;and exit
;

CE5B    JSR     &CDDA   ;exchange text column/line with workspace
CE5E    SEC             ;set carry
CE5F    LDA     &0309   ;bottom margin
CE62    SBC     &030B   ;top of text window
CE65    STA     &DE     ;store it
CE67    BNE     &CE6E   ;set text column to left hand column
CE69    PLA             ;get back return address
CE6A    PLA             ;
CE6B    JMP     &CDDA   ;exchange text column/line with workspace

CE6E    LDA     &0308   ;text window left
CE71    BPL     &CEE3   ;Jump CEE3 always!

CE73    LDA     &DA     ;get back A
CE75    PHA             ;push A
CE76    SEC             ;set carry
CE77    LDA     &030A   ;text window right
CE7A    SBC     &0308   ;text window left
CE7D    STA     &DF     ;
CE7F    LDY     &034F   ;bytes per character to set loop counter

CE82    DEY             ;copy loop
CE83    LDA     (&DA),Y ;
CE85    STA     (&D8),Y ;
CE87    DEY             ;
CE88    BPL     &CE83   ;

CE8A    LDX     #&02    ;X=2
CE8C    CLC             ;clear carry
CE8D    LDA     &D8,X   ;
CE8F    ADC     &034F   ;bytes per character
CE92    STA     &D8,X   ;
CE94    LDA     &D9,X   ;
CE96    ADC     #&00    ;
CE98    BPL     &CE9E   ;if this remains in screen RAM OK

CE9A    SEC             ;else wrap around screen
CE9B    SBC     &0354   ;screen RAM size hi byte
CE9E    STA     &D9,X   ;
CEA0    DEX             ;X=X-2
CEA1    DEX             ;
CEA2    BEQ     &CE8C   ;if X=0 adjust second set of pointers
CEA4    DEC     &DF     ;decrement window width
CEA6    BPL     &CE7F   ;and if still +ve do it all again
CEA8    PLA             ;get back A
CEA9    STA     &DA     ;and store it
CEAB    RTS             ;then exit
;
********** clear a line
*********************************************

CEAC    LDA     &0318   ;text column
CEAF    PHA             ;save it
CEB0    JSR     &CE6E   ;set text column to left hand column
CEB3    JSR     &CF06   ;set up display address
CEB6    SEC             ;set carry
CEB7    LDA     &030A   ;text window right
```

```
 CEBA    SBC     &0308   ;text window left
 CEBD    STA     &DC     ;as window width
 CEBF    LDA     &0358   ;background text colour
 CEC2    LDY     &034F   ;bytes per character

 CEC5    DEY             ;Y=Y-1 decrementing loop counter
 CEC6    STA     (&D8),Y ;store background colour at this point on screen
 CEC8    BNE     &CEC5   ;if Y<>0 do it again
 CECA    TXA             ;else A=X
 CECB    CLC             ;clear carry to add
 CECC    ADC     &034F   ;bytes per character
 CECF    TAX             ;X=A restoring it
 CED0    LDA     &D9     ;get hi byte
 CED2    ADC     #&00    ;Add carry if any
 CED4    BPL     &CEDA   ;if +ve CeDA
 CED6    SEC             ;else wrap around
 CED7    SBC     &0354   ;screen RAM size hi byte




 CEDA    STX     &D8     ;restore D8/9
 CEDC    STA     &D9     ;
 CEDE    DEC     &DC     ;decrement window width
 CEE0    BPL     &CEBF   ;ind if not 0 do it all again
 CEE2    PLA             ;get back A
 CEE3    STA     &0318   ;restore text column
 CEE6    SEC             ;set carry
 CEE7    RTS             ;and exit
 ;

 CEE8    LDX     &0318   ;text column
 CEEB    CPX     &0308   ;text window left
 CEEE    BMI     &CEE6   ;if less than left margin return with carry set
 CEF0    CPX     &030A   ;text window right
 CEF3    BEQ     &CEF7   ;if equal to right margin thats OK
 CEF5    BPL     &CEE6   ;if greater than right margin return with carry
set

 CEF7    LDX     &0319   ;current text line
 CEFA    CPX     &030B   ;top of text window
 CEFD    BMI     &CEE6   ;if less than top margin
 CEFF    CPX     &0309   ;bottom margin
 CF02    BEQ     &CF06   ;set up display address
 CF04    BPL     &CEE6   ;or greater than bottom margin return with carry
set


 ************:set up display address
************************************

 ;Mode 0: (0319)*640+(0318)* 8
 ;Mode 1: (0319)*640+(0318)*16
 ;Mode 2: (0319)*640+(0318)*32
 ;Mode 3: (0319)*640+(0318)* 8
 ;Mode 4: (0319)*320+(0318)* 8
 ;Mode 5: (0319)*320+(0318)*16
```

```
 ;Mode 6: (0319)*320+(0318)* 8
 ;Mode 7: (0319)* 40+(0318)
 ;this gives a displacement relative to the screen RAM start address
 ;which is added to the calculated number and stored in in 34A/B
 ;if the result is less than &8000, the top of screen RAM it is copied
into X/A
 ;and D8/9.
 ;if the result is greater than &7FFF the hi byte of screen RAM size is
 ;subtracted to wraparound the screen. X/A, D8/9 are then set from this

 CF06    LDA     &0319   ;current text line
 CF09    ASL             ;A=A*2
 CF0A    TAY             ;Y=A
 CF0B    LDA     (&E0),Y ;get CRTC multiplication table pointer
 CF0D    STA     &D9     ;&D9=A
 CF0F    INY             ;Y=Y+1
 CF10    LDA     #&02    ;A=2
 CF12    AND     &0356   ;memory map type
 CF15    PHP             ;save flags
 CF16    LDA     (&E0),Y ;get CRTC multiplication table pointer
 CF18    PLP             ;pull flags
 CF19    BEQ     &CF1E   ;
 CF1B    LSR     &D9     ;&D9=&D9/2
 CF1D    ROR             ;A=A/2 +(128*carry)
 CF1E    ADC     &0350   ;window area start address lo
 CF21    STA     &D8     ;store it
 CF23    LDA     &D9     ;
 CF25    ADC     &0351   ;window area start address hi
 CF28    TAY             ;
 CF29    LDA     &0318   ;text column
 CF2C    LDX     &034F   ;bytes per character
 CF2F    DEX             ;X=X-1
 CF30    BEQ     &CF44   ;if X=0 mode 7 CF44
 CF32    CPX     #&0F    ;is it mode 1 or mode 5?
 CF34    BEQ     &CF39   ;yes CF39 with carry set
 CF36    BCC     &CF3A   ;if its less (mode 0,3,4,6) CF3A
 CF38    ASL             ;A=A*16 if entered here (mode 2)

 CF39    ASL             ;A=A*8 if entered here

 CF3A    ASL             ;A=A*4 if entered here
 CF3B    ASL             ;
 CF3C    BCC     &CF40   ;if carry clear
 CF3E    INY             ;Y=Y+2
 CF3F    INY             ;
 CF40    ASL             ;A=A*2
 CF41    BCC     &CF45   ;if carry clear add to &D8
 CF43    INY             ;if not Y=Y+1

 CF44    CLC             ;clear carry
 CF45    ADC     &D8     ;add to &D8
 CF47    STA     &D8     ;and store it
 CF49    STA     &034A   ;text cursor 6845 address
 CF4C    TAX             ;X=A
 CF4D    TYA             ;A=Y
 CF4E    ADC     #&00    ;Add carry if set
 CF50    STA     &034B   ;text cursor 6845 address
 CF53    BPL     &CF59   ;if less than &800 goto &CF59
 CF55    SEC             ;else wrap around
 CF56    SBC     &0354   ;screen RAM size hi byte
```

```
 CF59    STA     &D9     ;store in high byte
 CF5B    CLC             ;clear carry
 CF5C    RTS             ;and exit


 ******** Graphics cursor display routine
*******************************

 CF5D    LDX     &0359   ;foreground graphics colour
 CF60    LDY     &035B   ;foreground graphics plot mode (GCOL n)
 CF63    JSR     &D0B3   ;set graphics byte mask in &D4/5
 CF66    JSR     &D486   ;copy (324/7) graphics cursor to workspace
(328/B)
 CF69    LDY     #&00    ;Y=0
 CF6B    STY     &DC     ;&DC=Y
 CF6D    LDY     &DC     ;Y=&DC
 CF6F    LDA     (&DE),Y ;get pattern byte
 CF71    BEQ     &CF86   ;if A=0 CF86
 CF73    STA     &DD     ;else &DD=A
 CF75    BPL     &CF7A   ;and if >0 CF7A
 CF77    JSR     &D0E3   ;else display a pixel
 CF7A    INC     &0324   ;current horizontal graphics cursor
 CF7D    BNE     &CF82   ;
 CF7F    INC     &0325   ;current horizontal graphics cursor

 CF82    ASL     &DD     ;&DD=&DD*2
 CF84    BNE     &CF75   ;and if<>0 CF75
 CF86    LDX     #&28    ;point to workspace
 CF88    LDY     #&24    ;point to horizontal graphics cursor
 CF8A    JSR     &D482   ;0300/1+Y=0300/1+X
 CF8D    LDY     &0326   ;current vertical graphics cursor
 CF90    BNE     &CF95   ;
 CF92    DEC     &0327   ;current vertical graphics cursor
 CF95    DEC     &0326   ;current vertical graphics cursor
 CF98    LDY     &DC     ;
 CF9A    INY             ;
 CF9B    CPY     #&08    ;if Y<8 then do loop again
 CF9D    BNE     &CF6B   ;else
 CF9F    LDX     #&28    ;point to workspace
 CFA1    LDY     #&24    ;point to graphics cursor
 CFA3    JMP     &D48A   ;(&300/3+Y)=(&300/3+X)


 *********** home graphics cursor ************************************

 CFA6    LDX     #&06    ;point to graphics window TOP
 CFA8    LDY     #&26    ;point to workspace
 CFAA    JSR     &D482   ;0300/1+Y=0300/1+X


 ************* set graphics cursor to left hand column
******************

 CFAD    LDX     #&00    ;X=0 point to graphics window left
 CFAF    LDY     #&24    ;Y=&24
 CFB1    JSR     &D482   ;0300/1+Y=0300/1+X
 CFB4    JMP     &D1B8   ;set up external coordinates for graphics
 CFB7    LDX     &0360   ;number of logical colours less 1
 CFBA    BEQ     &CFDC   ;if MODE 7 CFDC
```

```
 CFBC    JSR     &D03E   ;set up character definition pointers
 CFBF    LDX     &0360   ;number of logical colours less 1
 CFC2    LDA     &D0     ;VDU status byte
 CFC4    AND     #&20    ;and out bit 5 printing at graphics cursor
 CFC6    BNE     &CF5D   ;if set CF5D
 CFC8    LDY     #&07    ;else Y=7
 CFCA    CPX     #&03    ;if X=3
 CFCC    BEQ     &CFEE   ;goto CFEE to handle 4 colour modes
 CFCE    BCS     &D01E   ;else if X>3 D01E to deal with 16 colours

 CFD0    LDA     (&DE),Y ;get pattern byte
 CFD2    ORA     &D2     ;text colour byte to be orred or EORed into
memory
 CFD4    EOR     &D3     ;text colour byte to be orred or EORed into
memory
 CFD6    STA     (&D8),Y ; write to screen
 CFD8    DEY             ;Y=Y-1
 CFD9    BPL     &CFD0   ;if still +ve do loop again
 CFDB    RTS             ;and exit

 ******* convert teletext characters
************************************
 ;mode 7
 CFDC    LDY     #&02    ;Y=2
 CFDE    CMP     &C4B6,Y ;compare with teletext conversion table
 CFE1    BEQ     &CFE9   ;if equal then CFE9
 CFE3    DEY             ;else Y=Y-1
 CFE4    BPL     &CFDE   ;and if +ve CFDE

 CFE6    STA     (&D8,X) ;if not write byte to screen
 CFE8    RTS             ;and exit



 CFE9    LDA     &C4B7,Y ;convert with teletext conversion table
 CFEC    BNE     &CFE6   ;and write it


 ***********four colour modes
*****************************************

 CFEE    LDA     (&DE),Y ;get pattern byte
 CFF0    PHA             ;save it
 CFF1    LSR             ;move hi nybble to lo
 CFF2    LSR             ;
 CFF3    LSR             ;
 CFF4    LSR             ;
 CFF5    TAX             ;X=A
 CFF6    LDA     &C31F,X ;4 colour mode byte mask look up table
 CFF9    ORA     &D2     ;text colour byte to be orred or EORed into
memory
 CFFB    EOR     &D3     ;text colour byte to be orred or EORed into
memory
 CFFD    STA     (&D8),Y ; write to screen
 CFFF    TYA             ;A=Y

 D000    CLC             ;clear carry
 D001    ADC     #&08    ;add 8 to move screen RAM pointer 8 bytes
 D003    TAY             ;Y=A
```

```
 D004    PLA             ;get back A
 D005    AND     #&0F    ;clear high nybble
 D007    TAX             ;X=A
 D008    LDA     &C31F,X ;4 colour mode byte mask look up table
 D00B    ORA     &D2     ;text colour byte to be orred or EORed into
memory
 D00D    EOR     &D3     ;text colour byte to be orred or EORed into
memory
 D00F    STA     (&D8),Y ; write to screen
 D011    TYA             ;A=Y
 D012    SBC     #&08    ;A=A-9
 D014    TAY             ;Y=A
 D015    BPL     &CFEE   ;if +ve do loop again
 D017    RTS             ;exit



 D018    TYA             ;Y=Y-&21
 D019    SBC     #&21    ;
 D01B    BMI     &D017   ;IF Y IS negative then RETURN
 D01D    TAY             ;else A=Y

 ******* 16 COLOUR MODES
***********************************************

 D01E    LDA     (&DE),Y ;get pattern byte
 D020    STA     &DC     ;store it
 D022    SEC             ;set carry
 D023    LDA     #&00    ;A=0
 D025    ROL     &DC     ;carry now occupies bit 0 of DC
 D027    BEQ     &D018   ;when DC=0 again D018 to deal with next pattern
byte
 D029    ROL             ;get bit 7 from &DC into A bit 0
 D02A    ASL     &DC     ;rotate again to get second
 D02C    ROL             ;bit into A
 D02D    TAX             ;and store result in X
 D02E    LDA     &C32F,X ;multiply by &55 using look up table
 D031    ORA     &D2     ;and set colour factors
 D033    EOR     &D3     ;
 D035    STA     (&D8),Y ;and store result
 D037    CLC             ;clear carry
 D038    TYA             ;Y=Y+8 moving screen RAM pointer on 8 bytes
 D039    ADC     #&08    ;
 D03B    TAY             ;
 D03C    BCC     &D023   ;iloop to D023 to deal with next bit pair


 ************* calculate pattern address for given code
******************
 ;A contains code on entry = 12345678

 D03E    ASL             ;23456780  C holds 1
 D03F    ROL             ;34567801  C holds 2
 D040    ROL             ;45678012  C holds 3
 D041    STA     &DE     ;save this pattern
 D043    AND     #&03    ;00000012
 D045    ROL             ;00000123 C=0
 D046    TAX             ;X=A=0 - 7
 D047    AND     #&03    ;A=00000023
 D049    ADC     #&BF    ;A=&BF,C0 or C1
```

```
 D04B    TAY             ;this is used as a pointer
 D04C    LDA     &C40D,X ;A=&80/2^X i.e.1,2,4,8,&10,&20,&40, or &80
 D04F    BIT     &0367   ;with font flag
 D052    BEQ     &D057   ;if 0 D057
 D054    LDY     &0367,X ;else get hi byte from table
 D057    STY     &DF     ;store Y
 D059    LDA     &DE     ;get back pattern
 D05B    AND     #&F8    ;convert to 45678000
 D05D    STA     &DE     ;and re store it
 D05F    RTS             ;exit
                         ;

*************************************************************************

*************************************************************************
 **
**
 **
**
 **      PLOT ROUTINES ENTER HERE
**
 **
**
 **
**

*************************************************************************

*************************************************************************
 ;on entry    ADDRESS    PARAMETER        DESCRIPTION
 ;               031F    1               plot type
 ;               0320/1  2,3             X coordinate
 ;               0322/3  4,5             Y coordinate

 D060    LDX     #&20    ;X=&20
 D062    JSR     &D14D   ;translate xoordinates

 D065    LDA     &031F   ;get plot type
 D068    CMP     #&04    ;if its 4
 D06A    BEQ     &D0D9   ;D0D9 move absolute
 D06C    LDY     #&05    ;Y=5
 D06E    AND     #&03    ;mask only bits 0 and 1
 D070    BEQ     &D080   ;if result is 0 then its a move (multiple of 8)
 D072    LSR             ;else move bit 0 int C
 D073    BCS     &D078   ;if set then D078 graphics colour required
 D075    DEY             ;Y=4
 D076    BNE     &D080   ;logic inverse colour must be wanted

 ******** graphics colour wanted
****************************************

 D078    TAX             ;X=A if A=0 its a foreground colour 1 its
background
 D079    LDY     &035B,X ;get fore or background graphics PLOT mode
 D07C    LDA     &0359,X ;get fore or background graphics colour
 D07F    TAX             ;X=A

 D080    JSR     &D0B3   ;set up colour masks in D4/5

 D083    LDA     &031F   ;get plot type
```

```
 D086    BMI     &D0AB   ;if &80-&FF then D0AB type not implemented
 D088    ASL             ;bit 7=bit 6
 D089    BPL     &D0C6   ;if bit 6 is 0 then plot type is 0-63 so D0C6
 D08B    AND     #&F0    ;else mask out lower nybble
 D08D    ASL             ;shift old bit 6 into C bit old 5 into bit 7
 D08E    BEQ     &D0D6   ;if 0 then type 64-71 was called single point
plot
                         ;goto D0D6
 D090    EOR     #&40    ;if bit 6 NOT set type &80-&87 fill triangle
 D092    BEQ     &D0A8   ;so D0A8
 D094    PHA             ;else push A
 D095    JSR     &D0DC   ;copy 0320/3 to 0324/7 setting XY in current
graphics
                         ;coordinates
 D098    PLA             ;get back A
 D099    EOR     #&60    ;if BITS 6 and 5 NOT SET type 72-79 lateral fill
 D09B    BEQ     &D0AE   ;so D0AE
 D09D    CMP     #&40    ;if type 88-95 horizontal line blanking
 D09F    BNE     &D0AB   ;so D0AB

 D0A1    LDA     #&02    ;else A=2
 D0A3    STA     &DC     ;store it
 D0A5    JMP     &D506   ;and jump to D506 type not implemented

 D0A8    JMP     &D5EA   ;to fill triangle routine

 D0AB    JMP     &C938   ;VDU extension access entry

 D0AE    STA     &DC     ;store A
 D0B0    JMP     &D4BF   ;

 *********:set colour masks
**********************************************
 ;graphics mode in Y
 ;colour in X

 D0B3    TXA             ;A=X
 D0B4    ORA     &C41C,Y ;or with GCOL plot options table byte
 D0B7    EOR     &C41D,Y ;EOR with following byte
 D0BA    STA     &D4     ;and store it
 D0BC    TXA             ;A=X
 D0BD    ORA     &C41B,Y ;
 D0C0    EOR     &C420,Y ;
 D0C3    STA     &D5     ;
 D0C5    RTS             ;exit with masks in &D4/5


 ************** analyse first parameter in 0-63 range
********************
                         ;
 D0C6    ASL             ;shift left again
 D0C7    BMI     &D0AB   ;if -ve options are in range 32-63 not
implemented
 D0C9    ASL             ;shift left twice more
 D0CA    ASL             ;
 D0CB    BPL     &D0D0   ;if still +ve type is 0-7 or 16-23 so D0D0
 D0CD    JSR     &D0EB   ;else display a point

 D0D0    JSR     &D1ED   ;perform calculations
 D0D3    JMP     &D0D9   ;
```

```
*************************************************************************
 *
*
 *       PLOT A SINGLE POINT
*
 *
*

*************************************************************************

 D0D6    JSR     &D0EB   ;display a point
 D0D9    JSR     &CDE2   ;swap current and last graphics position
 D0DC    LDY     #&24    ;Y=&24
 D0DE    LDX     #&20    ;X=&20
 D0E0    JMP     &D48A   ;copy parameters to 324/7 (300/3 +Y)


 D0E3    LDX     #&24    ;
 D0E5    JSR     &D85F   ;calculate position
 D0E8    BEQ     &D0F0   ;if result =0 then D0F0
 D0EA    RTS             ;else exit
                         ;
 D0EB    JSR     &D85D   ;calculate position
 D0EE    BNE     &D103   ;if A<>0 D103 and return
 D0F0    LDY     &031A   ;else get current graphics scan line
 D0F3    LDA     &D1     ;pick up and modify screen byte
 D0F5    AND     &D4     ;
 D0F7    ORA     (&D6),Y ;
 D0F9    STA     &DA     ;
 D0FB    LDA     &D5     ;
 D0FD    AND     &D1     ;
 D0FF    EOR     &DA     ;
 D101    STA     (&D6),Y ;put it back again
 D103    RTS             ;and exit
                         ;

 D104    LDA     (&D6),Y ;this is a more simplistic version of the above
 D106    ORA     &D4     ;
 D108    EOR     &D5     ;
 D10A    STA     (&D6),Y ;
 D10C    RTS             ;and exit


 ************************* Check window limits ************************
                         ;
 D10D    LDX     #&24    ;X=&24
 D10F    LDY     #&00    ;Y=0
 D111    STY     &DA     ;&DA=0
 D113    LDY     #&02    ;Y=2
 D115    JSR     &D128   ;check vertical graphics position 326/7
                         ;bottom and top margins 302/3, 306/7
 D118    ASL     &DA     ;DATA is set in &DA bits 0 and 1 then shift left
 D11A    ASL     &DA     ;twice to make room for next pass
 D11C    DEX             ;X=&22
 D11D    DEX             ;
 D11E    LDY     #&00    ;Y=0
 D120    JSR     &D128   ;left and right margins 300/1, 304/5
```

```
                        ;cursor horizontal position 324/5
 D123    INX            ;X=X+2
 D124    INX            ;
 D125    LDA     &DA    ;A=&DA
 D127    RTS            ;exit


 *** cursor and margins check ***************************************
                        ;
 D128    LDA     &0302,X ;check for window violation
 D12B    CMP     &0300,Y ;300/1 +Y > 302/3+X
 D12E    LDA     &0303,X ;then window fault
 D131    SBC     &0301,Y ;
 D134    BMI     &D146   ;so D146

 D136    LDA     &0304,Y ;check other windows
 D139    CMP     &0302,X ;
 D13C    LDA     &0305,Y ;
 D13F    SBC     &0303,X ;
 D142    BPL     &D148   ;if no violation exit
 D144    INC     &DA     ;else DA=DA+1

 D146    INC     &DA     ;DA=DA+1
 D148    RTS             ;and exit  DA=0 no problems DA=1 first check 2,
2nd
                        ;


 ***********set up and adjust positional data
****************************

 D149    LDA     #&FF    ;A=&FF
 D14B    BNE     &D150   ;then &D150

 D14D    LDA     &031F   ;get first parameter in plot

 D150    STA     &DA     ;store in &DA
 D152    LDY     #&02    ;Y=2
 D154    JSR     &D176   ;set up vertical coordinates/2
 D157    JSR     &D1AD   ;/2 again to convert 1023 to 0-255 for internal
use
                        ;this is why minimum vertical plot separation is
4
 D15A    LDY     #&00    ;Y=0
 D15C    DEX             ;X=x-2
 D15D    DEX             ;
 D15E    JSR     &D176   ;set up horiz. coordinates/2 this is OK for
mode0,4
 D161    LDY     &0361   ;get number of pixels/byte (-1)
 D164    CPY     #&03    ;if Y=3 (modes 1 and 5)
 D166    BEQ     &D16D   ;D16D
 D168    BCS     &D170   ;for modes 0 & 4 this is 7 so D170
 D16A    JSR     &D1AD   ;for other modes divide by 2 twice

 D16D    JSR     &D1AD   ;divide by 2
 D170    LDA     &0356   ;get screen display type
 D173    BNE     &D1AD   ;if not 0 (modes 3-7) divide by 2 again
 D175    RTS             ;and exit

 ;for mode 0 1 division  1280 becomes 640 = horizontal resolution
```

```
;for mode 1 2 divisions 1280 becomes 320 = horizontal resolution
;for mode 2 3 divisions 1280 becomes 160 = horizontal resolution
;for mode 4 2 divisions 1280 becomes 320 = horizontal resolution
;for mode 5 3 divisions 1280 becomes 160 = horizontal resolution


********** calculate external coordinates in internal format ***********
;on entry X is usually &1E or &20

                        ;
D176    CLC             ;clear carry
D177    LDA     &DA     ;get &DA
D179    AND     #&04    ;if bit 2=0
D17B    BEQ     &D186   ;then D186 to calculate relative coordinates
D17D    LDA     &0302,X ;else get coordinate
D180    PHA             ;
D181    LDA     &0303,X ;
D184    BCC     &D194   ;and goto D194

D186    LDA     &0302,X ;get coordinate
D189    ADC     &0310,Y ;add cursor position
D18C    PHA             ;save it
D18D    LDA     &0303,X ;
D190    ADC     &0311,Y ;add cursor
D193    CLC             ;clear carry

D194    STA     &0311,Y ;save new cursor
D197    ADC     &030D,Y ;add graphics origin
D19A    STA     &0303,X ;store it
D19D    PLA             ;get back lo byte
D19E    STA     &0310,Y ;save it in new cursor lo
D1A1    CLC             ;clear carry
D1A2    ADC     &030C,Y ;add to graphics orgin
D1A5    STA     &0302,X ;store it
D1A8    BCC     &D1AD   ;if carry set
D1AA    INC     &0303,X ;increment hi byte as you would expect!

D1AD    LDA     &0303,X ;get hi byte
D1B0    ASL             ;
D1B1    ROR     &0303,X ;divide by 2
D1B4    ROR     &0302,X ;
D1B7    RTS             ;and exit
                        ;

***** calculate external coordinates from internal
coordinates************
D1B8    LDY     #&10    ;Y=10
D1BA    JSR     &D488   ;copy 324/7 to 310/3 i.e.current graphics cursor
                        ;position to position in external values
D1BD    LDX     #&02    ;X=2
D1BF    LDY     #&02    ;Y=2
D1C1    JSR     &D1D5   ;multiply 312/3 by 4 and subtract graphics
origin
D1C4    LDX     #&00    ;X=0
D1C6    LDY     #&04    ;Y=4
D1C8    LDA     &0361   ;get  number of pixels/byte
D1CB    DEY             ;Y=Y-1
D1CC    LSR             ;divide by 2
D1CD    BNE     &D1CB   ;if result not 0 D1CB
D1CF    LDA     &0356   ;else get screen display type
```

```
 D1D2    BEQ     &D1D5   ;and if 0 D1D5
 D1D4    INY             ;

 D1D5    ASL     &0310,X ;multiply coordinate by 2
 D1D8    ROL     &0311,X ;
 D1DB    DEY             ;Y-Y-1
 D1DC    BNE     &D1D5   ;and if Y<>0 do it again
 D1DE    SEC             ;set carry
 D1DF    JSR     &D1E3   ;
 D1E2    INX             ;increment X

 D1E3    LDA     &0310,X ;get current graphics position in external
coordinates
 D1E6    SBC     &030C,X ;subtract origin
 D1E9    STA     &0310,X ;store in graphics position
 D1EC    RTS             ;and exit
                         ;


 ************* compare X and Y PLOT spans
*********************************

 D1ED    JSR     &D40D   ;Set X and Y spans in workspace 328/9 32A/B
 D1F0    LDA     &032B   ;compare spans
 D1F3    EOR     &0329   ;if result -ve spans are different in sign so
 D1F6    BMI     &D207   ;goto D207
 D1F8    LDA     &032A   ;else A=hi byte of difference in spans
 D1FB    CMP     &0328   ;
 D1FE    LDA     &032B   ;
 D201    SBC     &0329   ;
 D204    JMP     &D214   ;and goto D214

 D207    LDA     &0328   ;A = hi byte of SUM of spans
 D20A    CLC             ;
 D20B    ADC     &032A   ;
 D20E    LDA     &0329   ;
 D211    ADC     &032B   ;


 D214    ROR             ;A=A/2
 D215    LDX     #&00    ;X=0
 D217    EOR     &032B   ;
 D21A    BPL     &D21E   ;if positive result D21E

 D21C    LDX     #&02    ;else X=2

 D21E    STX     &DE     ;store it
 D220    LDA     &C4AA,X ;set up vector address
 D223    STA     &035D   ;in 35D
 D226    LDA     &C4AB,X ;
 D229    STA     &035E   ;and 35E
 D22C    LDA     &0329,X ;get hi byte of span
 D22F    BPL     &D235   ;if +ve D235
 D231    LDX     #&24    ;X=&24
 D233    BNE     &D237   ;jump to D237

 D235    LDX     #&20    ;X=&20
 D237    STX     &DF     ;store it
 D239    LDY     #&2C    ;Y=&2C
 D23B    JSR     &D48A   ;get X coordinate data or horizontal coord of
```

```
                        ;curent graphics cursor
D23E    LDA     &DF     ;get back original X
D240    EOR     #&04    ;covert &20 to &24 and vice versa
D242    STA     &DD     ;
D244    ORA     &DE     ;
D246    TAX             ;
D247    JSR     &D480   ;copy 330/1 to 300/1+X
D24A    LDA     &031F   ;get plot type
D24D    AND     #&10    ;check bit 4
D24F    ASL             ;
D250    ASL             ;
D251    ASL             ;move to bit 7
D252    STA     &DB     ;store it
D254    LDX     #&2C    ;X=&2C
D256    JSR     &D10F   ;check for window violations
D259    STA     &DC     ;
D25B    BEQ     &D263   ;if none then D263
D25D    LDA     #&40    ;else set bit 6 of &DB
D25F    ORA     &DB     ;
D261    STA     &DB     ;

D263    LDX     &DD     ;
D265    JSR     &D10F   ;check window violations again
D268    BIT     &DC     ;if bit 7 of &DC NOT set
D26A    BEQ     &D26D   ;D26D
D26C    RTS             ;else exit
                        ;
D26D    LDX     &DE     ;X=&DE
D26F    BEQ     &D273   ;if X=0 D273
D271    LSR             ;A=A/2
D272    LSR             ;A=A/2

D273    AND     #&02    ;clear all but bit 2
D275    BEQ     &D27E   ;if bit 2 set D27E
D277    TXA             ;else A=X
D278    ORA     #&04    ;A=A or 4 setting bit 3
D27A    TAX             ;X=A
D27B    JSR     &D480   ;set 300/1+x to 330/1
D27E    JSR     &D42C   ;more calcualtions
D281    LDA     &DE     ;A=&DE EOR 2
D283    EOR     #&02    ;
D285    TAX             ;X=A
D286    TAY             ;Y=A
D287    LDA     &0329   ;compare upper byte of spans
D28A    EOR     &032B   ;
D28D    BPL     &D290   ;if signs are the same D290
D28F    INX             ;else X=X+1
D290    LDA     &C4AE,X ;get vector addresses and store 332/3
D293    STA     &0332   ;
D296    LDA     &C4B2,X ;
D299    STA     &0333   ;

D29C    LDA     #&7F    ;A=&7F
D29E    STA     &0334   ;store it
D2A1    BIT     &DB     ;if bit 6 set
D2A3    BVS     &D2CE   ;the D2CE
D2A5    LDA     &C447,X ;get VDU section number
D2A8    TAX             ;X=A
D2A9    SEC             ;set carry
D2AA    LDA     &0300,X ;subtract coordinates
```

```
D2AD    SBC     &032C,Y ;
D2B0    STA     &DA     ;
D2B2    LDA     &0301,X ;
D2B5    SBC     &032D,Y ;
D2B8    LDY     &DA     ;Y=hi
D2BA    TAX             ;X=lo=A
D2BB    BPL     &D2C0   ;and if A+Ve D2C0
D2BD    JSR     &D49B   ;negate Y/A

D2C0    TAX             ;X=A increment Y/A
D2C1    INY             ;Y=Y+1
D2C2    BNE     &D2C5   ;
D2C4    INX             ;X=X+1
D2C5    TXA             ;A=X
D2C6    BEQ     &D2CA   ;if A=0 D2CA
D2C8    LDY     #&00    ;else Y=0

D2CA    STY     &DF     ;&DF=Y
D2CC    BEQ     &D2D7   ;if 0 then D2D7
D2CE    TXA             ;A=X
D2CF    LSR             ;A=A/4
D2D0    ROR             ;
D2D1    ORA     #&02    ;bit 1 set
D2D3    EOR     &DE     ;
D2D5    STA     &DE     ;and store
D2D7    LDX     #&2C    ;X=&2C
D2D9    JSR     &D864   ;
D2DC    LDX     &DC     ;
D2DE    BNE     &D2E2   ;
D2E0    DEC     &DD     ;
D2E2    DEX             ;X=X-1
D2E3    LDA     &DB     ;A=&3B
D2E5    BEQ     &D306   ;if 0 D306
D2E7    BPL     &D2F9   ;else if +ve D2F9
D2E9    BIT     &0334   ;
D2EC    BPL     &D2F3   ;if bit 7=0 D2F3
D2EE    DEC     &0334   ;else decrement
D2F1    BNE     &D316   ;and if not 0 D316

D2F3    INC     &0334   ;
D2F6    ASL             ;A=A*2
D2F7    BPL     &D306   ;if +ve D306
D2F9    STX     &DC     ;
D2FB    LDX     #&2C    ;
D2FD    JSR     &D85F   ;calcualte screen position
D300    LDX     &DC     ;get back original X
D302    ORA     #&00    ;
D304    BNE     &D316   ;
D306    LDA     &D1     ;byte mask for current graphics point
D308    AND     &D4     ;and with graphics colour byte
D30A    ORA     (&D6),Y ;or  with curent graphics cell line
D30C    STA     &DA     ;store result
D30E    LDA     &D5     ;same again with next byte (hi??)
D310    AND     &D1     ;
D312    EOR     &DA     ;
D314    STA     (&D6),Y ;then store it inm current graphics line
D316    SEC             ;set carry
D317    LDA     &0335   ;A=&335/6-&337/8
D31A    SBC     &0337   ;
D31D    STA     &0335   ;
```

```
 D320    LDA     &0336   ;
 D323    SBC     &0338   ;
 D326    BCS     &D339   ;if carry set D339
 D328    STA     &DA     ;
 D32A    LDA     &0335   ;
 D32D    ADC     &0339   ;
 D330    STA     &0335   ;
 D333    LDA     &DA     ;
 D335    ADC     &033A   ;
 D338    CLC             ;
 D339    STA     &0336   ;
 D33C    PHP             ;
 D33D    BCS     &D348   ;if carry clear jump to VDU routine else D348
 D33F    JMP     (&0332) ;


 ****** vertical scan module 1****************************************

 D342    DEY             ;Y=Y-1
 D343    BPL     &D348   ;if + D348
 D345    JSR     &D3D3   ;else d3d3 to advance pointers
 D348    JMP     (&035D) ;and JUMP (&35D)


 ****** vertical scan module 2****************************************

 D34B    INY             ;Y=Y+1
 D34C    CPY     #&08    ;if Y<>8
 D34E    BNE     &D348   ;then D348
 D350    CLC             ;else clear carry
 D351    LDA     &D6     ;get address of top line of cuirrent graphics
cell
 D353    ADC     &0352   ;add number of bytes/character row
 D356    STA     &D6     ;store it
 D358    LDA     &D7     ;do same for hibyte
 D35A    ADC     &0353   ;
 D35D    BPL     &D363   ;if result -ve then we are above screen RAM
 D35F    SEC             ;so
 D360    SBC     &0354   ;subtract screen memory size hi
 D363    STA     &D7     ;store it this wraps around point to screen RAM
 D365    LDY     #&00    ;Y=0
 D367    JMP     (&035D) ;

 ****** horizontal scan module
1****************************************

 D36A    LSR     &D1     ;shift byte mask
 D36C    BCC     &D348   ;if carry clear (&D1 was +ve) goto D348
 D36E    JSR     &D3ED   ;else reset pointers
 D371    JMP     (&035D) ;and off to do more

 ****** horizontal scan module
2****************************************

 D374    ASL     &D1     ;shift byte mask
 D376    BCC     &D348   ;if carry clear (&D1 was +ve) goto D348
 D378    JSR     &D3FD   ;else reset pointers
 D37B    JMP     (&035D) ;and off to do more

 D37E    DEY             ;Y=Y-1
```

```
 D37F    BPL     &D38D   ;if +ve D38D
 D381    JSR     &D3D3   ;advance pointers
 D384    BNE     &D38D   ;goto D38D normally
 D386    LSR     &D1     ;shift byte mask
 D388    BCC     &D38D   ;if carry clear (&D1 was +ve) goto D348
 D38A    JSR     &D3ED   ;else reset pointers
 D38D    PLP             ;pull flags
 D38E    INX             ;X=X+1
 D38F    BNE     &D395   ;if X>0 D395
 D391    INC     &DD     ;else increment &DD
 D393    BEQ     &D39F   ;and if not 0 D39F
 D395    BIT     &DB     ;else if BIT 6 = 1
 D397    BVS     &D3A0   ;goto D3A0
 D399    BCS     &D3D0   ;if BIT 7=1 D3D0
 D39B    DEC     &DF     ;else Decrement &DF
 D39D    BNE     &D3D0   ;and if not Zero D3D0
 D39F    RTS             ;else return
                         ;
 D3A0    LDA     &DE     ;A=&DE
 D3A2    STX     &DC     ;&DC=X
 D3A4    AND     #&02    ;clear all but bit 1
 D3A6    TAX             ;X=A
 D3A7    BCS     &D3C2   ;and if carry set goto D3C2
 D3A9    BIT     &DE     ;if Bit 7 of &DE =1
 D3AB    BMI     &D3B7   ;then D3B7
 D3AD    INC     &032C,X ;else increment
 D3B0    BNE     &D3C2   ;and if not 0 D3C2
 D3B2    INC     &032D,X ;else increment hi byte
 D3B5    BCC     &D3C2   ;and if carry clear D3C2
 D3B7    LDA     &032C,X ;esle A=32C,X
 D3BA    BNE     &D3BF   ;and if not 0 D3BF
 D3BC    DEC     &032D,X ;decrement hi byte
 D3BF    DEC     &032C,X ;decrement lo byte

 D3C2    TXA             ;A=X
 D3C3    EOR     #&02    ;invert bit 2
 D3C5    TAX             ;X=A
 D3C6    INC     &032C,X ;Increment 32C/D
 D3C9    BNE     &D3CE   ;
 D3CB    INC     &032D,X ;
 D3CE    LDX     &DC     ;X=&DC
 D3D0    JMP     &D2E3   ;jump to D2E3


 **********move display point up a line
*********************************
 D3D3    SEC             ;SET CARRY
 D3D4    LDA     &D6     ;subtract number of bytes/line from address of
 D3D6    SBC     &0352   ;top line of current graphics cell
 D3D9    STA     &D6     ;
 D3DB    LDA     &D7     ;
 D3DD    SBC     &0353   ;
 D3E0    CMP     &034E   ;compare with bottom of screen memory
 D3E3    BCS     &D3E8   ;if outside screen RAM
 D3E5    ADC     &0354   ;add screen memory size to wrap it around
 D3E8    STA     &D7     ;store in current address of graphics cell top
line
 D3EA    LDY     #&07    ;Y=7
 D3EC    RTS             ;and RETURN
```

```
                        ;
D3ED    LDA     &0362   ;get current left colour mask
D3F0    STA     &D1     ;store it
D3F2    LDA     &D6     ;get current top line of graphics cell
D3F4    ADC     #&07    ;ADD 7
D3F6    STA     &D6     ;
D3F8    BCC     &D3FC   ;
D3FA    INC     &D7     ;
D3FC    RTS             ;and return
                        ;

D3FD    LDA     &0363   ;get right colour mask
D400    STA     &D1     ;store it
D402    LDA     &D6     ;A=top line graphics cell low
D404    BNE     &D408   ;if not 0 D408
D406    DEC     &D7     ;else decrement hi byte

D408    SBC     #&08    ;subtract 9 (8 + carry)
D40A    STA     &D6     ;and store in low byte
D40C    RTS             ;return
                        ;

 ********:: coordinate subtraction
**************************************

D40D    LDY     #&28    ;X=&28
D40F    LDX     #&20    ;Y=&20
D411    JSR     &D418   ;
D414    INX             ;X=X+2
D415    INX             ;
D416    INY             ;Y=Y+2
D417    INY             ;

D418    SEC             ;set carry
D419    LDA     &0304,X ;subtract coordinates
D41C    SBC     &0300,X ;
D41F    STA     &0300,Y ;
D422    LDA     &0305,X ;
D425    SBC     &0301,X ;
D428    STA     &0301,Y ;
D42B    RTS             ;and return
                        ;


D42C    LDA     &DE     ;A=&DE
D42E    BNE     &D437   ;if A=0 D437
D430    LDX     #&28    ;X=&28
D432    LDY     #&2A    ;Y=&2A
D434    JSR     &CDDE   ;exchange 300/1+Y with 300/1+X
                        ;IN THIS CASE THE X AND Y SPANS!

D437    LDX     #&28    ;X=&28
D439    LDY     #&37    ;Y=&37
D43B    JSR     &D48A   ;copy &300/4+Y to &300/4+X
                        ;transferring X and Y spans in this case
D43E    SEC             ;set carry
D43F    LDX     &DE     ;X=&DE
D441    LDA     &0330   ;subtract 32C/D,X from 330/1
D444    SBC     &032C,X ;
D447    TAY             ;partial answer in Y
```

```
D448    LDA     &0331   ;
D44B    SBC     &032D,X ;
D44E    BMI     &D453   ;if -ve D453
D450    JSR     &D49B   ;else negate Y/A

D453    STA     &DD     ;store A
D455    STY     &DC     ;and Y
D457    LDX     #&35    ;X=&35
D459    JSR     &D467   ;get coordinates
D45C    LSR             ;
D45D    STA     &0301,X ;
D460    TYA             ;
D461    ROR             ;
D462    STA     &0300,X ;
D465    DEX             ;
D466    DEX             ;


D467    LDY     &0304,X ;
D46A    LDA     &0305,X ;
D46D    BPL     &D47B   ;if A is +ve RETURN
D46F    JSR     &D49B   ;else negate Y/A
D472    STA     &0305,X ;store back again
D475    PHA             ;
D476    TYA             ;
D477    STA     &0304,X ;
D47A    PLA             ;get back A
D47B    RTS             ;and exit
                        ;
D47C    LDA     #&08    ;A=8
D47E    BNE     &D48C   ;copy 8 bytes
D480    LDY     #&30    ;Y=&30
D482    LDA     #&02    ;A=2
D484    BNE     &D48C   ;copy 2 bytes
D486    LDY     #&28    ;copy 4 bytes from 324/7 to 328/B
D488    LDX     #&24    ;
D48A    LDA     #&04    ;


**********copy A bytes from 300,X to 300,Y **************************

D48C    STA     &DA     ;
D48E    LDA     &0300,X ;
D491    STA     &0300,Y ;
D494    INX             ;
D495    INY             ;
D496    DEC     &DA     ;
D498    BNE     &D48E   ;
D49A    RTS             ;and return
                        ;


************* negation routine
****************************************

D49B    PHA             ;save A
D49C    TYA             ;A=Y
D49D    EOR     #&FF    ;invert
D49F    TAY             ;Y=A
```

```
 D4A0    PLA             ;get back A
 D4A1    EOR     #&FF    ;invert
 D4A3    INY             ;Y=Y+1
 D4A4    BNE     &D4A9   ;if not 0 exit
 D4A6    CLC             ;else
 D4A7    ADC     #&01    ;add 1 to A
 D4A9    RTS             ;return
                         ;
 D4AA    JSR     &D85D   ;check window boundaries and set up screen
pointer
 D4AD    BNE     &D4B7   ;if A<>0 D4B7
 D4AF    LDA     (&D6),Y ;else get byte from current graphics cell
 D4B1    EOR     &035A   ;compare with current background colour
 D4B4    STA     &DA     ;store it
 D4B6    RTS             ;and RETURN
                         ;

 D4B7    PLA             ;get back return link
 D4B8    PLA             ;
 D4B9    INC     &0326   ;increment current graphics cursor vertical lo
 D4BC    JMP     &D545   ;
```

OS SERIES IV
GEOFF COX

```
*************************************************************************
 *
*
 *  LATERAL FILL ROUTINE
*
 *
*

*************************************************************************

 D4BF    JSR     &D4AA   ;check current screen state
 D4C2    AND     &D1     ;if A and &D1 <> 0 a plotted point has been
found
 D4C4    BNE     &D4B9   ;so D4B9
 D4C6    LDX     #&00    ;X=0
 D4C8    JSR     &D592   ;update pointers
 D4CB    BEQ     &D4FA   ;if 0 then D4FA
 D4CD    LDY     &031A   ;else Y=graphics scan line
 D4D0    ASL     &D1     ;
 D4D2    BCS     &D4D9   ;if carry set D4D9
 D4D4    JSR     &D574   ;else D574
 D4D7    BCC     &D4FA   ;if carry clear D4FA
 D4D9    JSR     &D3FD   ;else D3FD to pick up colour multiplier
 D4DC    LDA     (&D6),Y ;get graphics cell line
 D4DE    EOR     &035A   ;EOR with background colour
 D4E1    STA     &DA     ;and store
 D4E3    BNE     &D4F7   ;if not 0 D4F7
 D4E5    SEC             ;else set carry
 D4E6    TXA             ;A=X
 D4E7    ADC     &0361   ;add pixels/byte
 D4EA    BCC     &D4F0   ;and if carry clear D4F0
 D4EC    INC     &DB     ;else increment &DB
 D4EE    BPL     &D4F7   ;and if +ve D4F7

 D4F0    TAX             ;else X=A
 D4F1    JSR     &D104   ;display a pixel
 D4F4    SEC             ;set carry
 D4F5    BCS     &D4D9   ;goto D4D9


 D4F7    JSR     &D574   ;
 D4FA    LDY     #&00    ;Y=0
 D4FC    JSR     &D5AC   ;
 D4FF    LDY     #&20    ;
 D501    LDX     #&24    ;
 D503    JSR     &CDE6   ;exchange 300/3 +Y with 300/3+X
 D506    JSR     &D4AA   ;check screen pixel
 D509    LDX     #&04    ;Y=5
 D50B    JSR     &D592   ;
 D50E    TXA             ;A=x
 D50F    BNE     &D513   ;if A<>0 d513
 D511    DEC     &DB     ;else  &DB=&dB-1

 D513    DEX             ;X=X-1
 D514    JSR     &D54B   ;
 D517    BCC     &D540   ;
```

```
D519    JSR     &D3ED   ;update pointers
D51C    LDA     (&D6),Y ;get byte from graphics line
D51E    EOR     &035A   ;EOR with background colour
D521    STA     &DA     ;and store it
D523    LDA     &DC     ;
D525    BNE     &D514   ;If A-0 back to D514
D527    LDA     &DA     ;else A=&DA
D529    BNE     &D53D   ;if A<>d53D
D52B    SEC             ;else set carry
D52C    TXA             ;A=x
D52D    ADC     &0361   ;Add number of pixels/byte
D530    BCC     &D536   ;and if carry clear D536
D532    INC     &DB     ;else inc DB
D534    BPL     &D53D   ;and if +ve D53D
D536    TAX             ;get back X
D537    JSR     &D104   ;display a point
D53A    SEC             ;set carry
D53B    BCS     &D519   ;goto D519

D53D    JSR     &D54B   ;
D540    LDY     #&04    ;
D542    JSR     &D5AC   ;


D545    JSR     &D0D9   ;
D548    JMP     &D1B8   ;scale pointers

D54B    LDA     &D1     ;get byte mask
D54D    PHA             ;save it
D54E    CLC             ;clear carry
D54F    BCC     &D560   ;

D551    PLA             ;get back A
D552    INX             ;X=X+1
D553    BNE     &D559   ;if not 0 D559
D555    INC     &DB     ;else inc &DB
D557    BPL     &D56F   ;if +ve D56F
D559    LSR     &D1     ;
D55B    BCS     &D56F   ;if Bit 7 D1 set D56F
D55D    ORA     &D1     ;else or withA
D55F    PHA             ;save result
D560    LDA     &D1     ;A=&D1
D562    BIT     &DA     ;test bits 6 and 7 of &DA
D564    PHP             ;save flags
D565    PLA             ;get into A
D566    EOR     &DC     ;EOR and DC
D568    PHA             ;save A
D569    PLP             ;
D56A    BEQ     &D551   ;

D56C    PLA             ;A=A EOR &D1 (byte mask)
D56D    EOR     &D1     ;
D56F    STA     &D1     ;store it
D571    JMP     &D0F0   ;and display a pixel

D574    LDA     #&00    ;A=0
D576    CLC             ;Clear carry

D577    BCC     &D583   ;goto D583 if carry clear
```

```
D579    INX             ;X=X+1
D57A    BNE     &D580   ;If <>0 D580
D57C    INC     &DB     ;else inc &DB
D57E    BPL     &D56F   ;and if +ve d56F

D580    ASL             ;A=A*2
D581    BCS     &D58E   ;if C set D58E
D583    ORA     &D1     ;else A=A OR (&D1)
D585    BIT     &DA     ;set V and M from &DA b6 b7
D587    BEQ     &D579   ;
D589    EOR     &D1     ;A=AEOR &D1
D58B    LSR             ;/2
D58C    BCC     &D56F   ;if carry clear D56F
D58E    ROR             ;*2
D58F    SEC             ;set carry
D590    BCS     &D56F   ;to D56F


D592    LDA     &0300,X ;Y/A=(&300/1 +X)-(&320/1)
D595    SEC             ;
D596    SBC     &0320   ;
D599    TAY             ;
D59A    LDA     &0301,X ;
D59D    SBC     &0321   ;
D5A0    BMI     &D5A5   ;if result -ve D5A5
D5A2    JSR     &D49B   ;or negate Y/A
D5A5    STA     &DB     ;store A
D5A7    TYA             ;A=Y
D5A8    TAX             ;X=A
D5A9    ORA     &DB     ;
D5AB    RTS             ;exit


                        ;
D5AC    STY     &DA     ;Y=&DA
D5AE    TXA             ;A=X
D5AF    TAY             ;Y=A
D5B0    LDA     &DB     ;A=&DB
D5B2    BMI     &D5B6   ;if -ve D5B6
D5B4    LDA     #&00    ;A=0
D5B6    LDX     &DA     ;X=&DA
D5B8    BNE     &D5BD   ;if <>0 D5BD
D5BA    JSR     &D49B   ;negate
D5BD    PHA             ;
D5BE    CLC             ;
D5BF    TYA             ;
D5C0    ADC     &0300,X ;Y/A+(&300/1 +X)=(&320/1)
D5C3    STA     &0320   ;
D5C6    PLA             ;
D5C7    ADC     &0301,X ;
D5CA    STA     &0321   ;
D5CD    RTS             ;return



*********************************************************************
 *
*
 *       OSWORD 13 read last two graphic cursor positions
*
```

```
 *
*

*************************************************************************
                        ;
 D5CE    LDA     #&03    ;A=3
 D5D0    JSR     &D5D5   ;
 D5D3    LDA     #&07    ;A=7
 D5D5    PHA             ;Save A
 D5D6    JSR     &CDE2   ;exchange last 2 graphics cursor coordinates
with
                        ;current coordinates
 D5D9    JSR     &D1B8   ;convert to external coordinates
 D5DC    LDX     #&03    ;X=3
 D5DE    PLA             ;save A
 D5DF    TAY             ;Y=A
 D5E0    LDA     &0310,X ;get graphics coordinate
 D5E3    STA     (&F0),Y ;store it in OS buffer
 D5E5    DEY             ;decrement Y and X
 D5E6    DEX             ;
 D5E7    BPL     &D5E0   ;if +ve do it again
 D5E9    RTS             ;then Exit
                         ;



*************************************************************************
 *
*
 *       PLOT Fill triangle routine
*
 *
*

*************************************************************************


 D5EA    LDX     #&20    ;X=&20
 D5EC    LDY     #&3E    ;Y=&3E
 D5EE    JSR     &D47C   ;copy 300/7+X to 300/7+Y
                         ;this gets XY data parameters and current
graphics
                         ;cursor position
 D5F1    JSR     &D632   ;exchange 320/3 with 324/7 if 316/7=<322/3
 D5F4    LDX     #&14    ;X=&14
 D5F6    LDY     #&24    ;Y=&24
 D5F8    JSR     &D636   ;
 D5FB    JSR     &D632   ;

 D5FE    LDX     #&20    ;
 D600    LDY     #&2A    ;
 D602    JSR     &D411   ;calculate 032A/B-(324/5-320/1)
 D605    LDA     &032B   ;and store
 D608    STA     &0332   ;result

 D60B    LDX     #&28    ;set pointers
 D60D    JSR     &D459   ;
 D610    LDY     #&2E    ;

 D612    JSR     &D0DE   ;copy 320/3 32/31
```

```
 D615    JSR     &CDE2   ;exchange 314/7 with 324/7
 D618    CLC             ;
 D619    JSR     &D658   ;execute fill routine

 D61C    JSR     &CDE2   ;
 D61F    LDX     #&20    ;
 D621    JSR     &CDE4   ;
 D624    SEC             ;
 D625    JSR     &D658   ;

 D628    LDX     #&3E    ;;X=&3E
 D62A    LDY     #&20    ;;Y=&20
 D62C    JSR     &D47C   ;;copy 300/7+X to 300/7+Y
 D62F    JMP     &D0D9   ;;this gets XY data parameters and current
graphics
                         ;cursor position


 D632    LDX     #&20    ;X=&20
 D634    LDY     #&14    ;Y=&14
 D636    LDA     &0302,X ;
 D639    CMP     &0302,Y ;
 D63C    LDA     &0303,X ;
 D63F    SBC     &0303,Y ;
 D642    BMI     &D657   ;if 302/3+Y>302/3+X return
 D644    JMP     &CDE6   ;else swap 302/3+X with 302/3+Y

                         ;



*************************************************************************
 *
*
 *       OSBYTE 134  Read cursor position
*
 *
*

*************************************************************************


 D647    LDA     &0318   ;read current text cursor (X)
 D64A    SEC             ;set carry
 D64B    SBC     &0308   ;subtract left hand column of current text
window
 D64E    TAX             ;X=A
 D64F    LDA     &0319   ;get current text cursor (Y)
 D652    SEC             ;
 D653    SBC     &030B   ;suptract top row of current window
 D656    TAY             ;Y=A
 D657    RTS             ;and exit


                         ;PLOT routines continue
                         ;many of the following routines are just
manipulations
                         ;only points of interest will be explained
 D658    PHP             ;store flags
 D659    LDX     #&20    ;X=&20
```

```
D65B    LDY     #&35    ;Y=&35
D65D    JSR     &D411   ;335/6=(324/5+X-320/1)
D660    LDA     &0336   ;
D663    STA     &033D   ;
D666    LDX     #&33    ;
D668    JSR     &D459   ;set pointers

D66B    LDY     #&39    ;set 339/C=320/3
D66D    JSR     &D0DE   ;
D670    SEC             ;
D671    LDA     &0322   ;
D674    SBC     &0326   ;
D677    STA     &031B   ;
D67A    LDA     &0323   ;
D67D    SBC     &0327   ;
D680    STA     &031C   ;
D683    ORA     &031B   ;check VDU queue
D686    BEQ     &D69F   ;

D688    JSR     &D6A2   ;display a line
D68B    LDX     #&33    ;
D68D    JSR     &D774   ;update pointers
D690    LDX     #&28    ;
D692    JSR     &D774   ;and again!
D695    INC     &031B   ;update VDU queue
D698    BNE     &D688   ;and if not empty do it again
D69A    INC     &031C   ;else increment next byte
D69D    BNE     &D688   ;and do it again

D69F    PLP             ;pull flags
D6A0    BCC     &D657   ;if carry clear exit
D6A2    LDX     #&39    ;
D6A4    LDY     #&2E    ;
D6A6    STX     &DE     ;
D6A8    LDA     &0300,X ;is 300/1+x<300/1+Y
D6AB    CMP     &0300,Y ;
D6AE    LDA     &0301,X ;
D6B1    SBC     &0301,Y ;
D6B4    BMI     &D6BC   ;if so D6BC
D6B6    TYA             ;else A=Y
D6B7    LDY     &DE     ;Y=&DE
D6B9    TAX             ;X=A
D6BA    STX     &DE     ;&DE=X
D6BC    STY     &DF     ;&DF=Y
D6BE    LDA     &0300,Y ;
D6C1    PHA             ;
D6C2    LDA     &0301,Y ;
D6C5    PHA             ;
D6C6    LDX     &DF     ;
D6C8    JSR     &D10F   ;check for window violations
D6CB    BEQ     &D6DA   ;
D6CD    CMP     #&02    ;
D6CF    BNE     &D70E   ;
D6D1    LDX     #&04    ;
D6D3    LDY     &DF     ;
D6D5    JSR     &D482   ;
D6D8    LDX     &DF     ;
D6DA    JSR     &D864   ;set a screen address
D6DD    LDX     &DE     ;X=&DE
D6DF    JSR     &D10F   ;check for window violations
```

```
 D6E2    LSR             ;A=A/2
 D6E3    BNE     &D70E   ;if A<>0 then exit
 D6E5    BCC     &D6E9   ;else if C clear D6E9
 D6E7    LDX     #&00    ;
 D6E9    LDY     &DF     ;
 D6EB    SEC             ;
 D6EC    LDA     &0300,Y ;
 D6EF    SBC     &0300,X ;
 D6F2    STA     &DC     ;
 D6F4    LDA     &0301,Y ;
 D6F7    SBC     &0301,X ;
 D6FA    STA     &DD     ;
 D6FC    LDA     #&00    ;
 D6FE    ASL             ;
 D6FF    ORA     &D1     ;
 D701    LDY     &DC     ;
 D703    BNE     &D719   ;
 D705    DEC     &DD     ;
 D707    BPL     &D719   ;
 D709    STA     &D1     ;
 D70B    JSR     &D0F0   ;display a point
 D70E    LDX     &DF     ;restore X
 D710    PLA             ;and A
 D711    STA     &0301,X ;store it
 D714    PLA             ;get back A
 D715    STA     &0300,X ;and store it
 D718    RTS             ;exit
                         ;
 D719    DEC     &DC     ;
 D71B    TAX             ;
 D71C    BPL     &D6FE   ;
 D71E    STA     &D1     ;
 D720    JSR     &D0F0   ;display a point
 D723    LDX     &DC     ;
 D725    INX             ;
 D726    BNE     &D72A   ;
 D728    INC     &DD     ;
 D72A    TXA             ;
 D72B    PHA             ;
 D72C    LSR     &DD     ;
 D72E    ROR             ;
 D72F    LDY     &0361   ;number of pixels/byte
 D732    CPY     #&03    ;if 3 mode = goto D73B
 D734    BEQ     &D73B   ;
 D736    BCC     &D73E   ;else if <3 mode 2 goto D73E
 D738    LSR     &DD     ;else rotate bottom bit of &DD
 D73A    ROR             ;into Accumulator

 D73B    LSR     &DD     ;rotate bottom bit of &DD
 D73D    LSR             ;into Accumulator
 D73E    LDY     &031A   ;Y=line in current graphics cell containing
current
                         ;point
 D741    TAX             ;X=A
 D742    BEQ     &D753   ;
 D744    TYA             ;Y=Y-8
 D745    SEC             ;
 D746    SBC     #&08    ;
 D748    TAY             ;
```

```
 D749    BCS     &D74D   ;
 D74B    DEC     &D7     ;decrement byte of top line off current graphics
cell
 D74D    JSR     &D104   ;display a point
 D750    DEX             ;
 D751    BNE     &D744   ;
 D753    PLA             ;
 D754    AND     &0361   ;pixels/byte
 D757    BEQ     &D70E   ;
 D759    TAX             ;
 D75A    LDA     #&00    ;A=0
 D75C    ASL             ;
 D75D    ORA     &0363   ;or with right colour mask
 D760    DEX             ;
 D761    BNE     &D75C   ;
 D763    STA     &D1     ;store as byte mask
 D765    TYA             ;Y=Y-8
 D766    SEC             ;
 D767    SBC     #&08    ;
 D769    TAY             ;
 D76A    BCS     &D76E   ;if carry clear
 D76C    DEC     &D7     ;decrement byte of top line off current graphics
cell
 D76E    JSR     &D0F3   ;display a point
 D771    JMP     &D70E   ;and exit via D70E

 D774    INC     &0308,X ;
 D777    BNE     &D77C   ;
 D779    INC     &0309,X ;
 D77C    SEC             ;
 D77D    LDA     &0300,X ;
 D780    SBC     &0302,X ;
 D783    STA     &0300,X ;
 D786    LDA     &0301,X ;
 D789    SBC     &0303,X ;
 D78C    STA     &0301,X ;
 D78F    BPL     &D7C1   ;
 D791    LDA     &030A,X ;
 D794    BMI     &D7A1   ;
 D796    INC     &0306,X ;
 D799    BNE     &D7AC   ;
 D79B    INC     &0307,X ;
 D79E    JMP     &D7AC   ;
 D7A1    LDA     &0306,X ;
 D7A4    BNE     &D7A9   ;
 D7A6    DEC     &0307,X ;
 D7A9    DEC     &0306,X ;
 D7AC    CLC             ;
 D7AD    LDA     &0300,X ;
 D7B0    ADC     &0304,X ;
 D7B3    STA     &0300,X ;
 D7B6    LDA     &0301,X ;
 D7B9    ADC     &0305,X ;
 D7BC    STA     &0301,X ;
 D7BF    BMI     &D791   ;
 D7C1    RTS             ;
                         ;
                         ;
```

```
*********************************************************************
 *
*
 *       OSBYTE 135  Read character at text cursor position
*
 *
*

*********************************************************************

 D7C2    LDY     &0360   ;get number of logical colours
 D7C5    BNE     &D7DC   ;if Y<>0 mode <>7 so D7DC
 D7C7    LDA     (&D8),Y ;get address of top scan line of current text
chr
 D7C9    LDY     #&02    ;Y=2
 D7CB    CMP     &C4B7,Y ;compare with conversion table
 D7CE    BNE     &D7D4   ;if not equal D7d4
 D7D0    LDA     &C4B6,Y ;else get next lower byte from table
 D7D3    DEY             ;Y=Y-1
 D7D4    DEY             ;Y=Y-1
 D7D5    BPL     &D7CB   ;and if +ve do it again
 D7D7    LDY     &0355   ;Y=current screen mode
 D7DA    TAX             ;return with character in X
 D7DB    RTS             ;
                         ;
 D7DC    JSR     &D808   ;set up copy of the pattern bytes at text cursor
 D7DF    LDX     #&20    ;X=&20
 D7E1    TXA             ;A=&20
 D7E2    PHA             ;Save it
 D7E3    JSR     &D03E   ;get pattern address for code in A
 D7E6    PLA             ;get back A
 D7E7    TAX             ;and X
 D7E8    LDY     #&07    ;Y=7
 D7EA    LDA     &0328,Y ;get byte in pattern copy
 D7ED    CMP     (&DE),Y ;check against pattern source
 D7EF    BNE     &D7F9   ;if not the same D7F9
 D7F1    DEY             ;else Y=Y-1
 D7F2    BPL     &D7EA   ;and if +ve D7EA
 D7F4    TXA             ;A=X
 D7F5    CPX     #&7F    ;is X=&7F (delete)
 D7F7    BNE     &D7D7   ;if not D7D7
 D7F9    INX             ;else X=X+1
 D7FA    LDA     &DE     ;get byte lo address
 D7FC    CLC             ;clear carry
 D7FD    ADC     #&08    ;add 8
 D7FF    STA     &DE     ;store it
 D801    BNE     &D7E8   ;and go back to check next character if <>0

 D803    TXA             ;A=X
 D804    BNE     &D7E1   ;if <>0 D7E1
 D806    BEQ     &D7D7   ;else D7D7


 ***************** set up pattern copy
**********************************

 D808    LDY     #&07    ;Y=7

 D80A    STY     &DA     ;&DA=Y
```

```
 D80C    LDA     #&01    ;A=1
 D80E    STA     &DB     ;&DB=A
 D810    LDA     &0362   ;A=left colour mask
 D813    STA     &DC     ;store an &DC
 D815    LDA     (&D8),Y ;get a byte from current text character
 D817    EOR     &0358   ;EOR with text background colour
 D81A    CLC             ;clear carry
 D81B    BIT     &DC     ;and check bits of colour mask
 D81D    BEQ     &D820   ;if result =0 then D820
 D81F    SEC             ;else set carry
 D820    ROL     &DB     ;&DB=&DB+Carry
 D822    BCS     &D82E   ;if carry now set (bit 7 DB originally set) D82E
 D824    LSR     &DC     ;else  &DC=&DC/2
 D826    BCC     &D81B   ;if carry clear D81B
 D828    TYA             ;A=Y
 D829    ADC     #&07    ;ADD ( (7+carry)
 D82B    TAY             ;Y=A
 D82C    BCC     &D810   ;
 D82E    LDY     &DA     ;read modified values into Y and A
 D830    LDA     &DB     ;
 D832    STA     &0328,Y ;store copy
 D835    DEY             ;and do it again
 D836    BPL     &D80A   ;until 8 bytes copied
 D838    RTS             ;exit
                         ;
 ********* pixel reading
**********************************************

 D839    PHA             ;store A
 D83A    TAX             ;X=A
 D83B    JSR     &D149   ;set up positional data
 D83E    PLA             ;get back A
 D83F    TAX             ;X=A
 D840    JSR     &D85F   ;set a screen address after checking for window
                         ;violations
 D843    BNE     &D85A   ;if A<>0 D85A to exit with A=&FF
 D845    LDA     (&D6),Y ;else get top line of current graphics cell
 D847    ASL             ;A=A*2 C=bit 7
 D848    ROL     &DA     ;&DA=&DA+2 +C  C=bit 7 &DA
 D84A    ASL     &D1     ;byte mask=bM*2 +carry from &DA
 D84C    PHP             ;save flags
 D84D    BCS     &D851   ;if carry set D851
 D84F    LSR     &DA     ;else restore &DA with bit '=0
 D851    PLP             ;pull flags
 D852    BNE     &D847   ;if Z set D847
 D854    LDA     &DA     ;else A=&DA AND number of colours in current
mode -1
 D856    AND     &0360   ;
 D859    RTS             ;then exit
                         ;
 D85A    LDA     #&FF    ;A=&FF
 D85C    RTS             ;exit
                         ;


 **********: check for window violations and set up screen address
**********

 D85D    LDX     #&20    ;X=&20
 D85F    JSR     &D10F   ;
```

```
 D862    BNE     &D85C   ;if A<>0 there is a window violation so D85C
 D864    LDA     &0302,X ;else set up graphics scan line variable
 D867    EOR     #&FF    ;
 D869    TAY             ;
 D86A    AND     #&07    ;
 D86C    STA     &031A   ;in 31A
 D86F    TYA             ;A=Y
 D870    LSR             ;A=A/2
 D871    LSR             ;A=A/2
 D872    LSR             ;A=A/2
 D873    ASL             ;A=A*2 this gives integer value bit 0 =0
 D874    TAY             ;Y=A
 D875    LDA     (&E0),Y ;get high byte of offset from screen RAM start
 D877    STA     &DA     ;store it
 D879    INY             ;Y=Y+1
 D87A    LDA     (&E0),Y ;get lo byte
 D87C    LDY     &0356   ;get screen map type
 D87F    BEQ     &D884   ;if 0 (modes 0,1,2) goto D884
 D881    LSR     &DA     ;else &DA=&DA/2
 D883    ROR             ;and A=A/2 +C if set
                         ;so 2 byte offset =offset/2

 D884    ADC     &0350   ;add screen top left hand corner lo
 D887    STA     &D6     ;store it
 D889    LDA     &DA     ;get high  byte
 D88B    ADC     &0351   ;add top left hi
 D88E    STA     &D7     ;store it
 D890    LDA     &0301,X ;
 D893    STA     &DA     ;
 D895    LDA     &0300,X ;
 D898    PHA             ;
 D899    AND     &0361   ;and then Add pixels per byte-1
 D89C    ADC     &0361   ;
 D89F    TAY             ;Y=A
 D8A0    LDA     &C406,Y ;A=&80 /2^Y using look up table
 D8A3    STA     &D1     ;store it
 D8A5    PLA             ;get back A
 D8A6    LDY     &0361   ;Y=&number of pixels/byte
 D8A9    CPY     #&03    ;is Y=3 (modes 1,6)
 D8AB    BEQ     &D8B2   ;goto D8B2
 D8AD    BCS     &D8B5   ;if mode =1 or 4 D8B5
 D8AF    ASL             ;A/&DA =A/&DA *2
 D8B0    ROL     &DA     ;

 D8B2    ASL             ;
 D8B3    ROL     &DA     ;

 D8B5    AND     #&F8    ;clear bits 0-2
 D8B7    CLC             ;clear carry
 D8B8    ADC     &D6     ;add A/&DA to &D6/7
 D8BA    STA     &D6     ;
 D8BC    LDA     &DA     ;
 D8BE    ADC     &D7     ;
 D8C0    BPL     &D8C6   ;if result +ve D8C6
 D8C2    SEC             ;else set carry
 D8C3    SBC     &0354   ;and subtract screen memory size making it wrap
round

 D8C6    STA     &D7     ;store it in &D7
```

```
 D8C8    LDY     &031A   ;get line in graphics cell containing current
graphics
 D8CB    LDA     #&00    ;point  A=0
 D8CD    RTS             ;And exit
                         ;
 D8CE    PHA             ;Push A
 D8CF    LDA     #&A0    ;A=&A0
 D8D1    LDX     &026A   ;X=number of items in VDU queque
 D8D4    BNE     &D916   ;if not 0 D916
 D8D6    BIT     &D0     ;else check VDU status byte
 D8D8    BNE     &D916   ;if either VDU is disabled or plot to graphics
                         ;cursor enabled then D916
 D8DA    BVS     &D8F5   ;if cursor editing enabled D8F5
 D8DC    LDA     &035F   ;else get 6845 register start setting
 D8DF    AND     #&9F    ;clear bits 5 and 6
 D8E1    ORA     #&40    ;set bit 6 to modify last cursor size setting
 D8E3    JSR     &C954   ;change write cursor format
 D8E6    LDX     #&18    ;X=&18
 D8E8    LDY     #&64    ;Y=&64
 D8EA    JSR     &D482   ;set text input cursor from text output cursor
 D8ED    JSR     &CD7A   ;modify character at cursor poistion
 D8F0    LDA     #&02    ;A=2
 D8F2    JSR     &C59D   ;bit 1 of VDU status is set to bar scrolling


 D8F5    LDA     #&BF    ;A=&BF
 D8F7    JSR     &C5A8   ;bit 6 of VDU status =0
 D8FA    PLA             ;Pull A
 D8FB    AND     #&7F    ;clear hi bit (7)
 D8FD    JSR     &C4C0   ;entire VDU routine !!
 D900    LDA     #&40    ;A=&40
 D902    JMP     &C59D   ;exit


 D905    LDA     #&20    ;A=&20
 D907    BIT     &D0     ;if bit 6 cursor editing is set
 D909    BVC     &D8CB   ;
 D90B    BNE     &D8CB   ;or bit 5 is set exit &D8CB
 D90D    JSR     &D7C2   ;read a character from the screen
 D910    BEQ     &D917   ;if A=0 on return exit via D917
 D912    PHA             ;else store A
 D913    JSR     &C664   ;perform cursor right

 D916    PLA             ;restore A
 D917    RTS             ;and exit
                         ;
 D918    LDA     #&BD    ;zero bits 2 and 6 of VDU status
 D91A    JSR     &C5A8   ;
 D91D    JSR     &C951   ;set normal cursor
 D920    LDA     #&0D    ;A=&0D
 D922    RTS             ;and return
                         ;this is response of CR as end of edit line



*********************************************************************
 *
*
 *       OSBYTE 132  Read bottom of display RAM
*
```

```
 *
*
*********************************************************************
                        ;
 D923    LDX     &0355   ;get current screen mode


*********************************************************************
 *
*
 *       OSBYTE 133  Read lowest address for given mode
*
 *
*
*********************************************************************

 D926    TXA             ;A=X
 D927    AND     #&07    ;MOD 7!
 D929    TAY             ;Y=A
 D92A    LDX     &C440,Y ;X=get RAM size key
 D92D    LDA     &C45E,X ;A=high byte of start address
 D930    LDX     #&00    ;X=0
 D932    BIT     &028E   ;get available RAM
 D935    BMI     &D93E   ;if bit 7 set then 32k so D93E
 D937    AND     #&3F    ;AND A with &3F
 D939    CPY     #&04    ;if Y<4
 D93B    BCS     &D93E   ;then D93E
 D93D    TXA             ;else A=0 to return null value
 D93E    TAY             ;Y=A
 D93F    RTS             ;and return



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

 D940    DB      10,E3   ;&E310 = USERV
&200
 D942    DB      54,DC   ;&DC54 = BRKV
&202
 D944    DB      93,DC   ;&DC93 = IRQ1V
&204
 D946    DB      89,DE   ;&DE89 = IRQ2V
&206
 D948    DB      89,DF   ;&DF89 = CLIV
&208
 D94A    DB      72,E7   ;&E772 = BYTEV
&20A
```

```
 D94C    DB      EB,E7   ;&E7EB = WORDV
&20C
 D94E    DB      A4,E0   ;&E0A4 = WRCHV
&20E
 D950    DB      C5,DE   ;&DEC5 = RDCHV
&210
 D952    DB      7D,F2   ;&F27D = FILEV
&212
 D954    DB      8E,F1   ;&F18E = ARGSV
&214
 D956    DB      C9,F4   ;&F4C9 = BGETV
&216
 D958    DB      29,F5   ;&F529 = BPUTV
&218
 D95A    DB      A6,FF   ;&FFA6 = GBPBV
&21A
 D95C    DB      CA,F3   ;&F3CA = FINDV
&21C
 D95E    DB      B1,F1   ;&F1B1 = FSCV
&21E
 D960    DB      A6,FF   ;&FFA6 = EVNTV
&220
 D962    DB      A6,FF   ;&FFA6 = UPTV
&222
 D964    DB      A6,FF   ;&FFA6 = NETV
&224
 D966    DB      A6,FF   ;&FFA6 = VDUV
&226
 D968    DB      02,EF   ;&EF02 = KEYV
&228
 D96A    DB      B3,E4   ;&E4B3 = INSBV
&22A
 D96C    DB      64,E4   ;&E464 = REMVB
&22C
 D96E    DB      D1,E1   ;&E1D1 = CNPV
&22E
 D970    DB      A6,FF   ;&FFA6 = IND1V
&230
 D972    DB      A6,FF   ;&FFA6 = IND2V
&232
 D974    DB      A6,FF   ;&FFA6 = IND3V
&234


*************************************************************************
 *
*
 *       MOS VARIABLES DEFAULT SETTINGS
*
*************************************************************************

 *read/written by Osbytes &A6 to &FC
 *addresses &236 to &28C =address -&D740

 D976    DB      90,01   ;Mos variables address  (address to Add to
osbyte
                        ;number) in lo hi format (&190)
&236
```

```
 D978    DB      9F,0D   ;Rom pointer address for indirecting into ROMS
                         ;=09DF (lo hi format)
&238

 D97A    DB      A1,02   ;ROM information table address (&2A1)
&23A

 D97C    DB      2B,F0   ;Key translation table address (&F02B)
&23C

 D97E    DB      00,03   ;VDU variables start 0300
&23E

 D980    DB      00      ;CFS/Vertical sync Timeout counter
&240
 D981    DB      00      ;current input buffer number
&241
 D982    DB      FF      ;keyboard interrupt processing flag
&242
 D983    DB      00      ;primary OSHWM  (default page)
&243
 D984    DB      00      ;current OSHWM  (PAGE)
&244
 D985    DB      01      ;RS423 Mode
&245
 D986    DB      00      ;character defininition explosion switch
&246
 D987    DB      00      ;Filing system flag ROM=2 CFS=0
&247
 D988    DB      00      ;current Video ULA control register
&248
 D989    DB      00      ;current pallette setting
&249
 D98A    DB      00      ;number of ROM enabled at last BRK
&24A
 D98B    DB      FF      ;number of BASIC ROM
&24B
 D98C    DB      04      ;current ADC channel number
&24C
 D98D    DB      04      ;maximum ADC channel number
&24D
 D98E    DB      00      ;ADC conversion type 0 or 0C=12 bit 8=8 Bit
&24E
 D98F    DB      FF      ;RS423 busy flag (bit 7 = 0 =busy)
&24F


 D990    DB      56      ;curent ACIA control register setting
&250
 D991    DB      19      ;flash counter
&251
 D992    DB      19      ;mark period count
&252
 D993    DB      19      ;space period count
&253
 D994    DB      32      ;keyboard Auto-repeat delay
&254
 D995    DB      08      ;keyboard Auto-repeat rate
&255
```

```
 D996    DB      00      ;*EXEC file handle  (0 -not allocated)
&256
 D997    DB      00      ;*SPOOL file handle (0 -not allocated)
&257
 D998    DB      00      ;bit 0 Escape enable/disable
&258
                         ;bit 1 BREAK normal/clear memory
 D999    DB      00      ;Econet disable keyboard flag
&259
 D99A    DB      20      ;keyboard status        bit 3=1 shift pressed
&25A
                                                 bit 4=0 caps  lock
                                                 bit 5=0 shift lock
                                                 bit 6=1 control bit
                                                 bit 7=1 shift enabled
 D99B    DB      09      ;buffer space left at buffer full signal
&25B
 D99C    DB      00      ;RS423 input suppression flag
&25C
 D99D    DB      00      ;cassette/RS423 flag  (0=CFS, &40=RS423)
&25D
 D99E    DB      00      ;Econet OS call interception flag (bit 7)
&25E
 D99F    DB      00      ;Econet OSRDCH interception flag (bit 7)
&25F

 D9A0    DB      00      ;Econet OSWRCH interception flag (bit 7)
&260
 D9A1    DB      50      ;speech enable/disable flac (50/20)
&261
 D9A2    DB      00      ;sound output enable flag
&262
 D9A3    DB      03      ;BELL channel number
&263
 D9A4    DB      90      ;BELL amplitude/Envelope number
&264
 D9A5    DB      64      ;BELL frequency
&265
 D9A6    DB      06      ;BELL duration
&266
 D9A7    DB      81      ;bit 7=1 ignore start up message
&267
                         ;bit 0=1 ignore RFS !BOOT error

 D9A8    DB      00      ;length of KEY string
&268
 D9A9    DB      00      ;PRINT line counter
&269
 D9AA    DB      00      ;number of items in VDU queque (2s complement
&26A
                         ;of number required)
 D9AB    DB      09      ;TAB key value
&26B
 D9AC    DB      1B      ;ESCAPE Character
&26C

        ;The following are input buffer code interpretation bytes for
        ;keys returning the following keys C0-FF are available via
        ;keypads only!
        ;0=ignore key
```

```
         ;1=expand as normal key
         ;2-FF add to base for ASCII code

 D9AD    DB      01      ;C0-CF
&26D
 D9AE    DB      D0      ;D0-DF
&26E
 D9AF    DB      E0      ;E0-EF
&26F
 D9B0    DB      F0      ;F0-FF
&270
 D9B1    DB      01      ;80-8F
&271
 D9B2    DB      80      ;90-9F
&272
 D9B3    DB      90      ;A0-AF
&273
 D9B4    DB      00      ;B0-BF
&274


 D9B5    DB      00      ;ESCAPE key status (0=ESC, 1,=ASCII)
&275
 D9B6    DB      00      ;ESCAPE action
&276
 D9B7    DB      FF      ;USER 6522 Bit IRQ mask
&277
 D9B8    DB      FF      ;6850 ACIA Bit IRQ bit mask
&278
 D9B9    DB      FF      ;System 6522 IRQ bit mask
&279
 D9BA    DB      00      ;Tube prescence flag
&27A
 D9BB    DB      00      ;speech processor prescence flag
&27B
 D9BC    DB      00      ;character destination status
&27C
 D9BD    DB      00      ;cursor editing status
&27D

 ****************** Warm reset high water mark
**************************

 D9BE    DB      00      ;unused
&27E
 D9BF    DB      00      ;unused
&27F
 D9C0    DB      00      ;country code
&280
 D9C1    DB      00      ;user flag
&281
 D9C2    DB      64      ;serial ULA control register setting
&282
 D9C3    DB      05      ;current system clock store pointer
&283
 D9C4    DB      FF      ;soft key status (unstable)
&284
 D9C5    DB      01      ;printer destination
&285
```

```
 D9C6    DB      0A      ;printer ignore character
&286

 ********* COLD RESET High water mark
************************************

 D9C7    DB      00      ;user BREAK routine address JMP
&288
 D9C8    DB      00      ;user BREAK routine address lo
&288
 D9C9    DB      00      ;user BREAK routine address hi
&289
 D9CA    DB      00      ;unused
&28A
 D9CB    DB      00      ;unused
&28B
 D9CC    DB      FF      ;current language rom no.
&28C

 *********** RESET High Water mark for Power up
*****************************
 ;later flags dealt with in routines




*******************************************************************************
*

*******************************************************************************
*
 **
**
 **
**
 **      RESET (BREAK) ENTRY POINT
**
 **
**
 **      Power up Enter with nothing set, 6522 System VIA IER bits
**
 **      0 to 6 will be clear
**
 **
**
 **      BREAK IER bits 0 to 6 one or more will be set 6522 IER
**
 **      not reset by BREAK
**
 **
**

*******************************************************************************
*

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

```
 D9CD    LDA     #&40    ;set NMI first instruction to RTI
 D9CF    STA     &0D00   ;NMI ram start

 D9D2    SEI             ;disable interrupts just in case
 D9D3    CLD             ;clear decimal flag
 D9D4    LDX     #&FF    ;reset stack to where it should be
 D9D6    TXS             ;(&1FF)
 D9D7    LDA     &FE4E   ;read interupt enable register of the system VIA
 D9DA    ASL             ;shift bit 7 into carry
 D9DB    PHA             ;save what's left
 D9DC    BEQ     &D9E7   ;if Power up A=0 so D9E7
 D9DE    LDA     &0258   ;else if BREAK pressed read BREAK Action flags
(set by
                         ;*FX200,n)
 D9E1    LSR             ;divide by 2
 D9E2    CMP     #&01    ;if (bit 1 not set by *FX200)
 D9E4    BNE     &DA03   ;then &DA03
 D9E6    LSR             ;divide A by 2 again (A=0 if *FX200,2/3 else
A=n/4



 ********** clear store routine
****************************************

 D9E7    LDX     #&04    ;get page to start clearance from (4)
 D9E9    STX     &01     ;store it in ZP 01
 D9EB    STA     &00     ;store A at 00

 D9ED    TAY             ;and in Y to set loop counter

 D9EE    STA     (&00),Y ;clear store
 D9F0    CMP     &01     ;until address &01 =0
 D9F2    BEQ     &D9FD   ;
 D9F4    INY             ;increment pointer
 D9F5    BNE     &D9EE   ;if not zero loop round again
 D9F7    INY             ;else increment again (Y=1) this avoids
overwriting
                         ;RTI instruction at &D00
 D9F8    INX             ;increment X
 D9F9    INC     &01     ;increment &01
 D9FB    BPL     &D9EE   ;loop until A=&80 then exit
                         ;note that RAM addressing for 16k loops around
so
                         ;&4000=&00 hence checking &01 for 00.  This
avoids
                         ;overwriting zero page on BREAK


 D9FD    STX     &028E   ;writes marker for available RAM 40 =16k,80=32
 DA00    STX     &0284   ;write soft key consistency flag

 **+********** set up system VIA
****************************************

 DA03    LDX     #&0F    ;set PORT B to output on bits 0-3 Input 4-7
 DA05    STX     &FE42   ;
```

```
*************************************************************************
 *
*
 *        set addressable latch IC 32 for peripherals via PORT B
*
 *
*
 *       ;bit 3 set sets addressed latch high adds 8 to VIA address
*
 *       ;bit 3 reset sets addressed latch low
*
 *
*
 *       Peripheral              VIA bit 3=0             VIA bit 3=1
*
 *
*
 *       Sound chip              Enabled                 Disabled
*
 *       speech chip (RS)        Low                     High
*
 *       speech chip (WS)        Low                     High
*
 *       Keyboard Auto Scan      Disabled                Enabled
*
 *       C0 address modifier     Low                     High
*
 *       C1 address modifier     Low                     High
*
 *       Caps lock  LED          ON                      OFF
*
 *       Shift lock LED          ON                      OFF
*
 *
*
 *       C0 & C1 are involved with hardware scroll screen address
*
*************************************************************************

         ;X=&F on entry

 DA08    DEX             ;loop start
 DA09    STX     &FE40   ;write latch IC32
 DA0C    CPX     #&09    ;is it 9
 DA0E    BCS     &DA08   ;if so go back and do it again
                         ;X=8 at this point
                         ;Caps lock On, SHIFT lock undetermined
                         ;Keyboard Autoscan on
                         ;sound disabled (may still sound)
 DA10    INX             ;X=9
 DA11    TXA             ;A=X
 DA12    JSR     &F02A   ;interrogate keyboard
 DA15    CPX     #&80    ;for keyboard links 9-2 and CTRL key (1)
 DA17    ROR     &FC     ;rotate MSB into bit 7 of &FC
```

```
 DA19    TAX             ;get back value of X for loop
 DA1A    DEX             ;decrement it
 DA1B    BNE     &DA11   ;and if >0 do loop again
                         ; on exit if Carry set link 3 made
                         ;link 2 = bit 0 of &FC and so on
                         ;if CTRL pressed bit 7 of &FC=1
                         ;X=0
 DA1D    STX     &028D   ;clear last BREAK flag
 DA20    ROL     &FC     ;CTRL is now in carry &FC is keyboard links
 DA22    JSR     &EEEB   ;set LEDs carry on entry  bit 7 of A on exit
 DA25    ROR             ;get carry back into carry flag

 ****** set up page 2
************************************************

 DA26    LDX     #&9C    ;
 DA28    LDY     #&8D    ;
 DA2A    PLA             ;get back A from &D9DB
 DA2B    BEQ     &DA36   ;if A=0 power up reset so DA36 with X=&9C Y=&8D
 DA2D    LDY     #&7E    ;else Y=&7E
 DA2F    BCC     &DA42   ;and if not CTRL-BREAK DA42 WARM RESET
 DA31    LDY     #&87    ;else Y=&87 COLD RESET
 DA33    INC     &028D   ;&28D=1

 DA36    INC     &028D   ;&28D=&28D+1
 DA39    LDA     &FC     ;get keyboard links set
 DA3B    EOR     #&FF    ;invert
 DA3D    STA     &028F   ;and store at &28F
 DA40    LDX     #&90    ;X=&90



 **********: set up page 2
************************************************

         ;on entry        &28D=0 Warm reset, X=&9C, Y=&7E
                         ;&28D=1 Power up  , X=&90, Y=&8D
                         ;&28D=2 Cold reset, X=&9C, Y=&87

 DA42    LDA     #&00    ;A=0
 DA44    CPX     #&CE    ;zero &200+X to &2CD
 DA46    BCC     &DA4A   ;
 DA48    LDA     #&FF    ;then set &2CE to &2FF to &FF
 DA4A    STA     &0200,X ;
 DA4D    INX             ;
 DA4E    BNE     &DA44   ;
                         ;A=&FF X=0
 DA50    STA     &FE63   ;set port A of user via to all outputs (printer
out)

 DA53    TXA             ;A=0
 DA54    LDX     #&E2    ;X=&E2
 DA56    STA     &00,X   ;zero zeropage &E2 to &FF
 DA58    INX             ;
 DA59    BNE     &DA56   ;X=0

 DA5B    LDA     &D93F,Y ;copy data from &D93F+Y
 DA5E    STA     &01FF,Y ;to &1FF+Y
```

```
 DA61    DEY             ;until
 DA62    BNE     &DA5B   ;1FF+Y=&200

 DA64    LDA     #&62    ;A=&62
 DA66    STA     &ED     ;store in &ED
 DA68    JSR     &FB0A   ;set up ACIA
                         ;X=0
 ************** clear interrupt and enable registers of Both VIAs
********

 DA6B    LDA     #&7F    ;
 DA6D    INX             ;
 DA6E    STA     &FE4D,X ;
 DA71    STA     &FE6D,X ;
 DA74    DEX             ;
 DA75    BPL     &DA6E   ;

 DA77    CLI             ;briefly allow interrupts to clear anything
pending
 DA78    SEI             ;disallow again N.B. All VIA IRQs are disabled
 DA79    BIT     &FC     ;if bit 6=1 then JSR &F055 (normally 0)
 DA7B    BVC     &DA80   ;else DA80
 DA7D    JSR     &F055   ;F055 JMP (&FDFE) probably causes a BRK unless
                         ;hardware there redirects it.
                         ;
 DA80    LDX     #&F2    ;enable interrupts 1,4,5,6 of system VIA
 DA82    STX     &FE4E   ;
                         ;0      Keyboard enabled as needed
                         ;1      Frame sync pulse
                         ;4      End of A/D conversion
                         ;5      T2 counter (for speech)
                         ;6      T1 counter (10 mSec intervals)
                         ;
 DA85    LDX     #&04    ;set system VIA PCR
 DA87    STX     &FE4C   ;
                         ;CA1 to interrupt on negative edge (Frame sync)
                         ;CA2 Handshake output for Keyboard
                         ;CB1 interrupt on negative edge (end of
conversion)
                         ;CB2 Negative edge (Light pen strobe)
                         ;
 DA8A    LDA     #&60    ;set system VIA ACR
 DA8C    STA     &FE4B   ;
                         ;disable latching
                         ;disable shift register
                         ;T1 counter continuous interrupts
                         ;T2 counter timed interrupt

 DA8F    LDA     #&0E    ;set system VIA T1 counter (Low)
 DA91    STA     &FE46   ;
                         ;this becomes effective when T1 hi set

 DA94    STA     &FE6C   ;set user VIA PCR
                         ;CA1 interrupt on -ve edge (Printer Acknowledge)
                         ;CA2 High output (printer strobe)
                         ;CB1 Interrupt on -ve edge (user port)
                         ;CB2 Negative edge (user port)

 DA97    STA     &FEC0   ;set up A/D converter
                         ;Bits 0 & 1 determine channel selected
```

```
                        ;Bit 3=0 8 bit conversion bit 3=1 12 bit

 DA9A    CMP     &FE6C   ;read user VIA IER if = &0E then DAA2 chip
present
 DA9D    BEQ     &DAA2   ;so goto DAA2
 DA9F    INC     &0277   ;else increment user VIA mask to 0 to bar all
                         ;user VIA interrupts

 DAA2    LDA     #&27    ;set T1 (hi) to &27 this sets T1 to &270E (9998
uS)
 DAA4    STA     &FE47   ;or 10msec, interrupts occur every 10msec
therefore
 DAA7    STA     &FE45   ;

 DAAA    JSR     &EC60   ;clear the sound channels

 DAAD    LDA     &0282   ;read serial ULA control register
 DAB0    AND     #&7F    ;zero bit 7
 DAB2    JSR     &E6A7   ;and set up serial ULA

 DAB5    LDX     &0284   ;get soft key status flag
 DAB8    BEQ     &DABD   ;if 0 (keys OK) then DABD
 DABA    JSR     &E9C8   ;else reset function keys



*************************************************************************
 *
*
 *       Check sideways ROMS and make catalogue
*
 *
*

*************************************************************************

                        ;X=0
 DABD    JSR     &DC16   ;set up ROM latch and RAM copy to X
 DAC0    LDX     #&03    ;set X to point to offset in table
 DAC2    LDY     &8007   ;get copyright offset from ROM

                        ; DF0C = )C( BRK
 DAC5    LDA     &8000,Y ;get first byte
 DAC8    CMP     &DF0C,X ;compare it with table byte
 DACB    BNE     &DAFB   ;if not the same then goto DAFB
 DACD    INY             ;point to next byte
 DACE    DEX             ;(s)
 DACF    BPL     &DAC5   ;and if still +ve go back to check next byte

                        ;this point is reached if 5 bytes indicate valid
                        ;ROM (offset +4 in (C) string)



*************************************************************************
 * Check first 1k of each ROM against higher priority Roms to ensure
that*
 * there are no matches, if a match found ignore lower priority ROM
*
```

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

DAD1    LDX     &F4     ;get RAM copy of ROM No. in X
DAD3    LDY     &F4     ;and Y

DAD5    INY             ;increment Y to check
DAD6    CPY     #&10    ;if ROM 15 is current ROM
DAD8    BCS     &DAFF   ;if equal or more than 16 goto &DAFF
                        ;to store catalogue byte
DADA    TYA             ;else put Y in A
DADB    EOR     #&FF    ;invert it
DADD    STA     &FA     ;and store at &FA
DADF    LDA     #&7F    ;store &7F at
DAE1    STA     &FB     ;&FB to get address &7FFF-Y

DAE3    STY     &FE30   ;set new ROM
DAE6    LDA     (&FA),Y ;Get byte
DAE8    STX     &FE30   ;switch back to previous ROM
DAEB    CMP     (&FA),Y ;and compare with previous byte called
DAED    BNE     &DAD5   ;if not the same then go back and do it again
                        ;with next rom up
DAEF    INC     &FA     ;else increment &FA to point to new location
DAF1    BNE     &DAE3   ;if &FA<>0 then check next byte
DAF3    INC     &FB     ;else inc &FB
DAF5    LDA     &FB     ;and check that it doesn't exceed
DAF7    CMP     #&84    ;&84 (1k checked)
DAF9    BCC     &DAE3   ;then check next byte(s)

DAFB    LDX     &F4     ;X=(&F4)
DAFD    BPL     &DB0C   ;if +ve then &DB0C

DAFF    LDA     &8006   ;get rom type
DB02    STA     &02A1,X ;store it in catalogue
DB05    AND     #&8F    ;check for BASIC (bit 7 not set)
DB07    BNE     &DB0C   ;if not BASIC the DB0C
DB09    STX     &024B   ;else store X at BASIC pointer


DB0C    INX             ;increment X to point to next ROM
DB0D    CPX     #&10    ;is it 15 or less
DB0F    BCC     &DABD   ;if so goto &DABD for next ROM
```

```
os series V
GEOFF COX

*************************************************************************
 *
*
 *       Check SPEECH System
*
 *
*

*************************************************************************

                        ;X=&10
 DB11    BIT     &FE40  ;if bit 7 low then we have speec system fitted
 DB14    BMI     &DB27  ;else goto DB27

 DB16    DEC     &027B  ;(027B)=&FF to indicate speech present

 DB19    LDY     #&FF   ;Y=&FF
 DB1B    JSR     &EE7F  ;initialise speech generator
 DB1E    DEX            ;via this
 DB1F    BNE     &DB19  ;loop
                        ;X=0
 DB21    STX     &FE48  ;set T2 timer for speech
 DB24    STX     &FE49  ;


 *********** SCREEN SET UP ********************************************
                        ;X=0
 DB27    LDA     &028F  ;get back start up options (mode)
 DB2A    JSR     &C300  ;then jump to screen initialisation

 DB2D    LDY     #&CA   ;Y=&CA
 DB2F    JSR     &E4F1  ;to enter this in keyboard buffer
                        ;this enables the *KEY 10 facility

 ********* enter BREAK intercept with Carry Clear
************************

 DB32    JSR     &EAD9  ;check to see if BOOT address is set up if so
                        ;JMP to it

 DB35    JSR     &F140  ;set up cassette options
 DB38    LDA     #&81   ;test for tube to FIFO buffer 1
 DB3A    STA     &FEE0  ;
 DB3D    LDA     &FEE0  ;
 DB40    ROR            ;put bit 0 into carry
 DB41    BCC     &DB4D  ;if no tube then DB4D
 DB43    LDX     #&FF   ;else
 DB45    JSR     &F168  ;issue ROM service call &FF
                        ;to initialise TUBE system
 DB48    BNE     &DB4D  ;if not 0 on exit (Tube not initialised) DB4D
 DB4A    DEC     &027A  ;else set tube flag to show its active


 DB4D    LDY     #&0E   ;set current value of PAGE
 DB4F    LDX     #&01   ;issue claim absolute workspace call
 DB51    JSR     &F168  ;via F168
 DB54    LDX     #&02   ;send private workspace claim call
```

```
 DB56    JSR     &F168   ;via F168
 DB59    STY     &0243   ;set primary OSHWM
 DB5C    STY     &0244   ;set current OSHWM
 DB5F    LDX     #&FE    ;issue call for Tube to explode character set
etc.
 DB61    LDY     &027A   ;Y=FF if tube present else Y=0
 DB64    JSR     &F168   ;and make call via F168

 DB67    AND     &0267   ;if A=&FE and bit 7 of 0267 is set then continue
 DB6A    BPL     &DB87   ;else ignore start up message
 DB6C    LDY     #&02    ;output to screen
 DB6E    JSR     &DEA9   ;'BBC Computer ' message
 DB71    LDA     &028D   ;0=warm reset, anything else continue
 DB74    BEQ     &DB82   ;
 DB76    LDY     #&16    ;by checking length of RAM
 DB78    BIT     &028E   ;
 DB7B    BMI     &DB7F   ;and either
 DB7D    LDY     #&11    ;
 DB7F    JSR     &DEA9   ;finishing message with '16k' or '32k'
 DB82    LDY     #&1B    ;and two newlines
 DB84    JSR     &DEA9   ;

 *********: enter BREAK INTERCEPT ROUTINE WITH CARRY SET (call 1)
 DB87    SEC             ;
 DB88    JSR     &EAD9   ;look for break intercept jump do *TV etc
 DB8B    JSR     &E9D9   ;set up LEDs in accordance with keyboard status
 DB8E    PHP             ;save flags
 DB8F    PLA             ;and get back in A
 DB90    LSR             ;zero bits 4-7 and bits 0-2 bit 4 which was bit
7
 DB91    LSR             ;may be set
 DB92    LSR             ;
 DB93    LSR             ;
 DB94    EOR     &028F   ;or with start up options which may or may not
 DB97    AND     #&08    ;invert bit 4
 DB99    TAY             ;Y=A
 DB9A    LDX     #&03    ;make initialisation call if Y=0 on entry
 DB9C    JSR     &F168   ;RUN, EXEC or LOAD !BOOT file
 DB9F    BEQ     &DBBE   ;if a ROM accepts this call then DBBE
 DBA1    TYA             ;else put Y in A
 DBA2    BNE     &DBB8   ;if Y<>0 DBB8
 DBA4    LDA     #&8D    ;else set up standard cassete baud rates
 DBA6    JSR     &F135   ;via &F135

 DBA9    LDX     #&D2    ;
 DBAB    LDY     #&EA    ;
 DBAD    DEC     &0267   ;decrement ignore start up message flag
 DBB0    JSR     OSCLI   ;and execute */!BOOT
 DBB3    INC     &0267   ;restore start up message flag
 DBB6    BNE     &DBBE   ;if not zero then DBBE

 DBB8    LDA     #&00    ;else A=0
 DBBA    TAX             ;X=0
 DBBB    JSR     &F137   ;set tape speed

 ******** Preserve current language on soft RESET
************************

 DBBE    LDA     &028D   ;get last RESET Type
 DBC1    BNE     &DBC8   ;if not soft reset DBC8
```

```
 DBC3    LDX     &028C   ;else get current language ROM address
 DBC6    BPL     &DBE6   ;if +ve (language available) then skip search
routine



*************************************************************************
 *
*
 *       SEARCH FOR LANGUAGE TO ENTER (Highest priority)
*
 *
*

*************************************************************************

 DBC8    LDX     #&0F    ;set pointer to highest available rom

 DBCA    LDA     &02A1,X ;get rom type from map
 DBCD    ROL             ;put hi-bit into carry, bit 6 into bit 7
 DBCE    BMI     &DBE6   ;if bit 7 set then ROM has a language entry so
DBE6

 DBD0    DEX             ;else search for language until X=&ff
 DBD1    BPL     &DBCA   ;


 *************** check if tube present
***********************************

 DBD3    LDA     #&00    ;if bit 7 of tube flag is set BMI succeeds
 DBD5    BIT     &027A   ;and TUBE is connected else
 DBD8    BMI     &DC08   ;make error

 ********* no language error
*********************************************

 DBDA    BRK             ;
 DBDB    DB      &F9     ;error number
 DBDC    DB      'Language?'     ;message
 DBE5    BRK             ;


 DBE6    CLC             ;



*************************************************************************
 *
*
 *       OSBYTE 142 enter Language ROM at &8000
*
 *
*
 *       X=rom number C set if OSBYTE call clear if initialisation
*
 *
*
```

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

 DBE7    PHP             ;save flags
 DBE8    STX     &028C   ;put X in current ROM page
 DBEB    JSR     &DC16   ;select that ROM
 DBEE    LDA     #&80    ;A=128
 DBF0    LDY     #&08    ;Y=8
 DBF2    JSR     &DEAB   ;display text string held in ROM at &8008,Y
 DBF5    STY     &FD     ;save Y on exit (end of language string)
 DBF7    JSR     OSNEWL  ;two line feeds
 DBFA    JSR     OSNEWL  ;are output
 DBFD    PLP             ;then get back flags
 DBFE    LDA     #&01    ;A=1 required for language entry
 DC00    BIT     &027A   ;check if tube exists
 DC03    BMI     &DC08   ;and goto DC08 if it does
 DC05    JMP     &8000   ;else enter language at &8000


*************************************************************************
 *
*
 *       TUBE FOUND enter tube software
*
 *
*

*************************************************************************

 DC08    JMP     &0400   ;enter tube environment


*************************************************************************
 *
*
 *       OSRDRM entry point
*
 *
*
 *       get byte from PHROM or page ROM
*
 *       Y= rom number, address is in &F6/7
*

*************************************************************************

 DC0B    LDX     &F4     ;get current ROM number into X
 DC0D    STY     &F4     ;store new number in &F4
 DC0F    STY     &FE30   ;switch in ROM
 DC12    LDY     #&00    ;get current PHROM address
 DC14    LDA     (&F6),Y ;and get byte

 ******** Set up Sideways Rom latch and RAM copy
************************
                 ;on entry X=ROM number

 DC16    STX     &F4     ;RAM copy of rom latch
 DC18    STX     &FE30   ;write to rom latch
```

```
 DC1B    RTS             ;and return



*********************************************************************
*
*********************************************************************
*
 **
**
 **      MAIN IRQ Entry point
**
 **
**
 **
**

*********************************************************************
*

*********************************************************************
*
 ;ON ENTRY STACK contains        STATUS REGISTER,PCH,PCL
;

 DC1C    STA     &FC     ;save A
 DC1E    PLA             ;get back status (flags)
 DC1F    PHA             ;and save again
 DC20    AND     #&10    ;check if BRK flag set
 DC22    BNE     &DC27   ;if so goto DC27
 DC24    JMP     (&0204) ;else JMP (IRQ1V)



*********************************************************************
 *
*
 *               BRK handling routine
*
 *
*

*********************************************************************

 DC27    TXA             ;save X on stack
 DC28    PHA             ;
 DC29    TSX             ;get status pointer
 DC2A    LDA     &0103,X ;get Program Counter lo
 DC2D    CLD             ;
 DC2E    SEC             ;set carry
 DC2F    SBC     #&01    ;subtract 2 (1+carry)
 DC31    STA     &FD     ;and store it in &FD
 DC33    LDA     &0104,X ;get hi byte
 DC36    SBC     #&00    ;subtract 1 if necessary
 DC38    STA     &FE     ;and store in &FE
 DC3A    LDA     &F4     ;get currently active ROM
 DC3C    STA     &024A   ;and store it in &24A
 DC3F    STX     &F0     ;store stack pointer in &F0
```

```
 DC41    LDX     #&06    ;and issue ROM service call 6
 DC43    JSR     &F168   ;(User BRK) to roms
                         ;at this point &FD/E point to byte after BRK
                         ;ROMS may use BRK for their own purposes

 DC46    LDX     &028C   ;get current language
 DC49    JSR     &DC16   ;and activate it
 DC4C    PLA             ;get back original value of X
 DC4D    TAX             ;
 DC4E    LDA     &FC     ;get back original value of A
 DC50    CLI             ;allow interrupts
 DC51    JMP     (&0202) ;and JUMP via BRKV (normally into current
language)



*************************************************************************
 *
*
 *       DEFAULT BRK HANDLER
*
 *
*

*************************************************************************

 DC54    LDY     #&00    ;Y=0 to point to byte after BRK
 DC56    JSR     &DEB1   ;print message

 DC59    LDA     &0267   ;if BIT 0 set and DISC EXEC error
 DC5C    ROR             ;occurs
 DC5D    BCS     &DC5D   ;hang up machine!!!!

 DC5F    JSR     OSNEWL  ;else print two newlines
 DC62    JSR     OSNEWL  ;
 DC65    JMP     &DBB8   ;and set tape speed before entering current
                         ;language

 DC68    SEC             ;set carry
 DC69    ROR     &024F   ;and rotate right to set RS423 busy flag
 DC6C    BIT     &0250   ;check bit 7 of current ACIA control register
 DC6F    BPL     &DC78   ;if interrupts NOT enabled DC78
 DC71    JSR     &E741   ;else E741 to check if serial buffer full
 DC74    LDX     #&00    ;
 DC76    BCS     &DC7A   ;if carry set goto DC7A to transfer data

 DC78    LDX     #&40    ;X=&40
 DC7A    JMP     &E17A   ;and transfer data

 DC7D    LDY     &FE09   ;read serial data from ACIA
 DC80    AND     #&3A    ;and %0011 1010
 DC82    BNE     &DCB8   ;if no 0 then DCB8

 DC84    LDX     &025C   ;read RS423 input suppression flag
 DC87    BNE     &DC92   ;if not 0 then DC92 ignore RS423 input
 DC89    INX             ;else X=X+1
 DC8A    JSR     &E4F3   ;put byte in buffer
 DC8D    JSR     &E741   ;count buffer
```

```
 DC90    BCC     &DC78   ;and if carry clear (buffer not full) back to
DC78
 DC92    RTS             ;else return
                         ;

*************************************************************************
 *
*
 *       Main IRQ Handling routines, default IRQIV destination
*
 *
*

*************************************************************************


 DC93    CLD             ;clear decimal flag
 DC94    LDA     &FC     ;get original value of A
 DC96    PHA             ;save it
 DC97    TXA             ;save X
 DC98    PHA             ;
 DC99    TYA             ;and Y
 DC9A    PHA             ;
 DC9B    LDA     #&DE    ;A=&DE
 DC9D    PHA             ;store it
 DC9E    LDA     #&81    ;save &81
 DCA0    PHA             ;store it (an RTS will now jump to DE82)
 DCA1    CLV             ;clear V flag
 DCA2    LDA     &FE08   ;get value of status register of ACIA
 DCA5    BVS     &DCA9   ;if parity error then DCA9
 DCA7    BPL     &DD06   ;else if no interrupt requested DD06

 DCA9    LDX     &EA     ;read RS423 timeout counter
 DCAB    DEX             ;decrement it
 DCAC    BMI     &DCDE   ;and if <0 DCDE
 DCAE    BVS     &DCDD   ;else if >&40 DCDD (RTS to DE82)
 DCB0    JMP     &F588   ;else read ACIA via F588

 DCB3    LDY     &FE09   ;read ACIA data
 DCB6    ROL             ;
 DCB7    ASL             ;
 DCB8    TAX             ;X=A
 DCB9    TYA             ;A=Y
 DCBA    LDY     #&07    ;Y=07
 DCBC    JMP     &E494   ;check and service EVENT 7 RS423 error

 DCBF    LDX     #&02    ;read RS423 output buffer
 DCC1    JSR     &E460   ;
 DCC4    BCC     &DCD6   ;if C=0 buffer is not empty goto DCD6
 DCC6    LDA     &0285   ;else read printer destination
 DCC9    CMP     #&02    ;is it serial printer??
 DCCB    BNE     &DC68   ;if not DC68
 DCCD    INX             ;else X=3
 DCCE    JSR     &E460   ;read printer buffer
 DCD1    ROR     &02D2   ;rotate to pass carry into bit 7
 DCD4    BMI     &DC68   ;if set then DC68
 DCD6    STA     &FE09   ;pass either printer or RS423 data to ACIA
 DCD9    LDA     #&E7    ;set timeout counter to stored value
 DCDB    STA     &EA     ;
 DCDD    RTS             ;and exit (to DE82)
```

```
                        ;A contains ACIA status
 DCDE    AND     &0278  ;AND with ACIA bit mask (normally FF)
 DCE1    LSR            ;rotate right to put bit 0 in carry
 DCE2    BCC     &DCEB  ;if carry clear receive register not full so
DCEB
 DCE4    BVS     &DCEB  ;if V is set then DCEB
 DCE6    LDY     &0250  ;else Y=ACIA control setting
 DCE9    BMI     &DC7D  ;if bit 7 set receive interrupt is enabled so
DC7D

 DCEB    LSR            ;put BIT 2 of ACIA status into
 DCEC    ROR            ;carry if set then Data Carrier Detected applies
 DCED    BCS     &DCB3  ;jump to DCB3

 DCEF    BMI     &DCBF  ;if original bit 1 is set TDR is empty so DCBF
 DCF1    BVS     &DCDD  ;if V is set then exit to DE82

 DCF3    LDX     #&05   ;X=5
 DCF5    JSR     &F168  ;issue rom call 5 'unrecognised interrupt'
 DCF8    BEQ     &DCDD  ;if a rom recognises it then exit to DE82
 DCFA    PLA            ;otherwise get back DE82 address from stack
 DCFB    PLA            ;
 DCFC    PLA            ;and get back X, Y, and A
 DCFD    TAY            ;
 DCFE    PLA            ;
 DCFF    TAX            ;
 DD00    PLA            ;
 DD01    STA     &FC    ;&FC=A
 DD03    JMP     (&0206) ;and offer to the user via IRQ2V



*************************************************************************
 *
*
 * VIA INTERUPTS ROUTINES
*
 *
*

*************************************************************************

 DD06    LDA     &FE4D  ;read system VIA interrupt flag register
 DD09    BPL     &DD47  ;if bit 7=0 the VIA has not caused interrupt
                        ;goto DD47
 DD0B    AND     &0279  ;mask with VIA bit mask
 DD0E    AND     &FE4E  ;and interrupt enable register
 DD11    ROR            ;rotate right twice to check for IRQ 1 (frame
sync)
 DD12    ROR            ;
 DD13    BCC     &DD69  ;if carry clear then no IRQ 1, else
 DD15    DEC     &0240  ;decrement vertical sync counter
 DD18    LDA     &EA    ;A=RS423 Timeout counter
 DD1A    BPL     &DD1E  ;if +ve then DD1E
 DD1C    INC     &EA    ;else increment it
 DD1E    LDA     &0251  ;load flash counter
 DD21    BEQ     &DD3D  ;if 0 then system is not in use, ignore it
```

```
 DD23    DEC     &0251   ;else decrement counter
 DD26    BNE     &DD3D   ;and if not 0 go on past reset routine

 DD28    LDX     &0252   ;else get mark period count in X
 DD2B    LDA     &0248   ;current VIDEO ULA control setting in A
 DD2E    LSR             ;shift bit 0 into C to check if first colour
 DD2F    BCC     &DD34   ;is effective if so C=0 jump to DD34

 DD31    LDX     &0253   ;else get space period count in X
 DD34    ROL             ;restore bit
 DD35    EOR     #&01    ;and invert it
 DD37    JSR     &EA00   ;then change colour

 DD3A    STX     &0251   ;&0251=X resetting the counter

 DD3D    LDY     #&04    ;Y=4 and call E494 to check and implement
vertical
 DD3F    JSR     &E494   ;sync event (4) if necessary
 DD42    LDA     #&02    ;A=2
 DD44    JMP     &DE6E   ;clear interrupt 1 and exit



*********************************************************************
 *
*
 *       PRINTER INTERRUPT USER VIA 1
*
 *
*

*********************************************************************

 DD47    LDA     &FE6D   ;Check USER VIA interrupt flags register
 DD4A    BPL     &DCF3   ;if +ve USER VIA did not call interrupt
 DD4C    AND     &0277   ;else check for USER IRQ 1
 DD4F    AND     &FE6E   ;
 DD52    ROR             ;
 DD53    ROR             ;
 DD54    BCC     &DCF3   ;if bit 1=0 the no interrupt 1 so DCF3
 DD56    LDY     &0285   ;else get printer type
 DD59    DEY             ;decrement
 DD5A    BNE     &DCF3   ;if not parallel then DCF3
 DD5C    LDA     #&02    ;reset interrupt 1 flag
 DD5E    STA     &FE6D   ;
 DD61    STA     &FE6E   ;disable interrupt 1
 DD64    LDX     #&03    ;and output data to parallel printer
 DD66    JMP     &E13A   ;


 *********************************************************************
 *
*
 *       SYSTEM INTERRUPT 5   Speech
*
 *
*

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

```
 DD69    ROL             ;get bit 5 into bit 7
 DD6A    ROL             ;
 DD6B    ROL             ;
 DD6C    ROL             ;
 DD6D    BPL     &DDCA   ;if not set the not a speech interrupt so DDCA

 DD6F    LDA     #&20    ;clear interrupt flag
 DD71    LDX     #&00    ;
 DD73    STA     &FE4D   ;
 DD76    STX     &FE49   ;and zer0 hi byte of T2 Timer
 DD79    LDX     #&08    ;&FB=8
 DD7B    STX     &FB     ;
 DD7D    JSR     &E45B   ;and examine buffer 8
 DD80    ROR     &02D7   ;shift carry into bit 7
 DD83    BMI     &DDC9   ;and if set buffer is empty so exit
 DD85    TAY             ;else Y=A
 DD86    BEQ     &DD8D   ;
 DD88    JSR     &EE6D   ;control speech chip
 DD8B    BMI     &DDC9   ;if negative exit
 DD8D    JSR     &E460   ;else get a byte from buffer
 DD90    STA     &F5     ;store it to indicate speech or file rom
 DD92    JSR     &E460   ;get another byte
 DD95    STA     &F7     ;store it
 DD97    JSR     &E460   ;and another
 DD9A    STA     &F6     ;giving address  to be accessed in paged ROM
 DD9C    LDY     &F5     ;Y=&F5
 DD9E    BEQ     &DDBB   ;and if =0 then DDBB
 DDA0    BPL     &DDB8   ;else if +ve DDB8
 DDA2    BIT     &F5     ;if bit 6 of F5 =1 (&F5)>&40
 DDA4    BVS     &DDAB   ;then DDAB
 DDA6    JSR     &EEBB   ;else continue for more speech processing
 DDA9    BVC     &DDB2   ;if bit 6 clear then DDB2
 DDAB    ASL     &F6     ;else double address in &F6/7
 DDAD    ROL     &F7     ;
 DDAF    JSR     &EE3B   ;and call EE3B
 DDB2    LDY     &0261   ;get speech enable/disable flag into Y
 DDB5    JMP     &EE7F   ;and JMP to EE7F

 DDB8    JSR     &EE7F   ;Call EE7F
 DDBB    LDY     &F6     ;get address pointer in Y
 DDBD    JSR     &EE7F   ;
 DDC0    LDY     &F7     ;get address pointer high in Y
 DDC2    JSR     &EE7F   ;
 DDC5    LSR     &FB     ;
 DDC7    BNE     &DD7D   ;
 DDC9    RTS             ;and exit
                         ;


 ************************************************************************
 *
*
 *       SYSTEM INTERRUPT 6 10mS Clock
*
 *
*

************************************************************************

 DDCA    BCC     &DE47   ;bit 6 is in carry so if clear there is no 6 int
```

```
                        ;so go on to DE47
 DDCC    LDA     #&40   ;Clear interrupt 6
 DDCE    STA     &FE4D  ;

 ;UPDATE timers routine, There are 2 timer stores &292-6 and &297-B
 ;these are updated by adding 1 to the current timer and storing the
 ;result in the other, the direction of transfer being changed each
 ;time of update.  This ensures that at least 1 timer is valid at any
call
 ;as the current timer is only read.  Other methods would cause
inaccuracies
 ;if a timer was read whilst being updated.

 DDD1    LDA     &0283  ;get current system clock store pointer (5,or
10)
 DDD4    TAX            ;put A in X
 DDD5    EOR     #&0F   ;and invert lo nybble (5 becomes 10 and vv)
 DDD7    PHA            ;store A
 DDD8    TAY            ;put A in Y

                        ;Carry is always set at this point
 DDD9    LDA     &0291,X ;get timer value
 DDDC    ADC     #&00   ;update it
 DDDE    STA     &0291,Y ;store result in alternate
 DDE1    DEX            ;decrement X
 DDE2    BEQ     &DDE7  ;if 0 exit
 DDE4    DEY            ;else decrement Y
 DDE5    BNE     &DDD9  ;and go back and do next byte

 DDE7    PLA            ;get back A
 DDE8    STA     &0283  ;and store back in clock pointer (i.e. inverse
previous
                        ;contents)
 DDEB    LDX     #&05   ;set loop pointer for countdown timer
 DDED    INC     &029B,X ;increment byte and if
 DDF0    BNE     &DDFA  ;not 0 then DDFA
 DDF2    DEX            ;else decrement pointer
 DDF3    BNE     &DDED  ;and if not 0 do it again
 DDF5    LDY     #&05   ;process EVENT 5 interval timer
 DDF7    JSR     &E494  ;

 DDFA    LDA     &02B1  ;get byte of inkey countdown timer
 DDFD    BNE     &DE07  ;if not 0 then DE07
 DDFF    LDA     &02B2  ;else get next byte
 DE02    BEQ     &DE0A  ;if 0 DE0A
 DE04    DEC     &02B2  ;decrement 2B2
 DE07    DEC     &02B1  ;and 2B1

 DE0A    BIT     &02CE  ;read bit 7 of envelope processing byte
 DE0D    BPL     &DE1A  ;if 0 then DE1A
 DE0F    INC     &02CE  ;else increment to 0
 DE12    CLI            ;allow interrupts
 DE13    JSR     &EB47  ;and do routine sound processes
 DE16    SEI            ;bar interrupts
 DE17    DEC     &02CE  ;DEC envelope processing byte back to 0

 DE1A    BIT     &02D7  ;read speech buffer busy flag
 DE1D    BMI     &DE2B  ;if set speech buffer is empty, skip routine
 DE1F    JSR     &EE6D  ;update speech system variables
 DE22    EOR     #&A0   ;
```

```
 DE24    CMP     #&60    ;
 DE26    BCC     &DE2B   ;if result >=&60 DE2B
 DE28    JSR     &DD79   ;else more speech work

 DE2B    BIT     &D9B7   ;set V and C
 DE2E    JSR     &DCA2   ;check if ACIA needs attention
 DE31    LDA     &EC     ;check if key has been pressed
 DE33    ORA     &ED     ;
 DE35    AND     &0242   ;(this is 0 if keyboard is to be ignored, else
&FF)
 DE38    BEQ     &DE3E   ;if 0 ignore keyboard
 DE3A    SEC             ;else set carry
 DE3B    JSR     &F065   ;and call keyboard
 DE3E    JSR     &E19B   ;check for data in user defined printer channel
 DE41    BIT     &FEC0   ;if ADC bit 6 is set ADC is not busy
 DE44    BVS     &DE4A   ;so DE4A
 DE46    RTS             ;else return
                         ;



*************************************************************************
 *
*
 *       SYSTEM INTERRUPT 4 ADC end of conversion
*
 *
*

*************************************************************************


 DE47    ROL             ;put original bit 4 from FE4D into bit 7 of A
 DE48    BPL     &DE72   ;if not set DE72

 DE4A    LDX     &024C   ;else get current ADC channel
 DE4D    BEQ     &DE6C   ;if 0 DE6C
 DE4F    LDA     &FEC2   ;read low data byte
 DE52    STA     &02B5,X ;store it in &2B6,7,8 or 9
 DE55    LDA     &FEC1   ;get high data byte
 DE58    STA     &02B9,X ;and store it in hi byte
 DE5B    STX     &02BE   ;store in Analogue system flag marking last
channel
 DE5E    LDY     #&03    ;handle event 3 conversion complete
 DE60    JSR     &E494   ;

 DE63    DEX             ;decrement X
 DE64    BNE     &DE69   ;if X=0
 DE66    LDX     &024D   ;get highest ADC channel preseny
 DE69    JSR     &DE8F   ;and start new conversion
 DE6C    LDA     #&10    ;reset interrupt 4
 DE6E    STA     &FE4D   ;
 DE71    RTS             ;and return



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

```
 *       SYSTEM INTERRUPT 0 Keyboard
*
 *
*

*********************************************************************

                          ;
 DE72    ROL              ;get original bit 0 in bit 7 position
 DE73    ROL              ;
 DE74    ROL              ;
 DE75    ROL              ;
 DE76    BPL     &DE7F    ;if bit 7 clear not a keyboard interrupt
 DE78    JSR     &F065    ;else scan keyboard
 DE7B    LDA     #&01     ;A=1
 DE7D    BNE     &DE6E    ;and off to reset interrupt and exit

 DE7F    JMP     &DCF3    ;


 ************** exit routine
*******************************************

 DE82    PLA              ;restore registers
 DE83    TAY              ;
 DE84    PLA              ;
 DE85    TAX              ;
 DE86    PLA              ;
 DE87    STA     &FC      ;store A



*********************************************************************
 *
*
 *       IRQ2V default entry
*
*********************************************************************

 DE89    LDA     &FC      ;get back original value of A
 DE8B    RTI              ;and return to calling routine




*********************************************************************
 *
*
 *       OSBYTE 17 Start conversion
*
 *
*

*********************************************************************

                          ;
 DE8C    STY     &02BE    ;set last channel to finish conversion
 DE8F    CPX     #&05     ;if X<4 then
 DE91    BCC     &DE95    ;DE95
```

```
DE93    LDX     #&04    ;else X=4

DE95    STX     &024C   ;store it as current ADC channel
DE98    LDY     &024E   ;get conversion type
DE9B    DEY             ;decrement
DE9C    TYA             ;A=Y
DE9D    AND     #&08    ;and it with 08
DE9F    CLC             ;clear carry
DEA0    ADC     &024C   ;add to current ADC
DEA3    SBC     #&00    ;-1
DEA5    STA     &FEC0   ;store to the A/D control panel
DEA8    RTS             ;and return
                        ;

DEA9    LDA     #&C3    ;point to start of string @&C300
DEAB    STA     &FE     ;store it
DEAD    LDA     #&00    ;point to lo byte
DEAF    STA     &FD     ;store it and start loop@

DEB1    INY             ;print character in string
DEB2    LDA     (&FD),Y ;pointed to by &FD/E
DEB4    JSR     OSASCI  ;print it expanding Carriage returns
DEB7    TAX             ;store A in X
DEB8    BNE     &DEB1   ;and loop again if not =0
DEBA    RTS             ;else exit


********** OSBYTE 129 TIMED ROUTINE ******************************
;ON ENTRY TIME IS IN X,Y

DEBB    STX     &02B1   ;store time in INKEY countdown timer
DEBE    STY     &02B2   ;which is decremented every 10ms
DEC1    LDA     #&FF    ;A=&FF
DEC3    BNE     &DEC7   ;goto DEC7



*************************************************************************
*

*************************************************************************
*
 **
**
 **      OSRDCH Default entry point
**
 **
**
 **      RDCHV entry point       read a character
**
 **
**

*************************************************************************
*

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

```
 DEC5    LDA     #&00    ;A=0

 DEC7    STA     &E6     ;store entry value of A
 DEC9    TXA             ;store X and Y
 DECA    PHA             ;
 DECB    TYA             ;
 DECC    PHA             ;
 DECD    LDY     &0256   ;get *EXEC file handle
 DED0    BEQ     &DEE6   ;if 0 (not allocated) then DEE6
 DED2    SEC             ;set carry
 DED3    ROR     &EB     ;set bit 7 of CFS active flag to prevent
clashes
 DED5    JSR     OSBGET  ;get a byte from the file
 DED8    PHP             ;push processor flags to preserve carry
 DED9    LSR     &EB     ;restore &EB
 DEDB    PLP             ;get back flags
 DEDC    BCC     &DF03   ;and if carry clear, character found so exit via
DF03
 DEDE    LDA     #&00    ;else A=00 as EXEC file empty
 DEE0    STA     &0256   ;store it in exec fil;e handle
 DEE3    JSR     OSFIND  ;and close file via OSFIND

 DEE6    BIT     &FF     ;check ESCAPE flag if bit 7 set Escape pressed
 DEE8    BMI     &DF00   ;so off to DF00
 DEEA    LDX     &0241   ;else get current input buffer number
 DEED    JSR     &E577   ;get a byte from keyboard buffer
 DEF0    BCC     &DF03   ;and exit if valid character found

 DEF2    BIT     &E6     ;(E6=0 or FF)
 DEF4    BVC     &DEE6   ;if entry was OSRDCH not timed keypress go back
and
                         ;do it again i.e. perform GET function
 DEF6    LDA     &02B1   ;else check timers
 DEF9    ORA     &02B2   ;
 DEFC    BNE     &DEE6   ;and if not zero go round again
 DEFE    BCS     &DF05   ;else exi
```

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
```

OS SERIES VII
GEOFF COX

```
*************************************************************************
 *
*
 *       OSBYTE  9   Duration of first colour
*
 *
*

*************************************************************************
 ;on entry Y=0, X=new value

                          ;
 E6B0    INY              ;Y is incremented to 1
 E6B1    CLC              ;clear carry



*************************************************************************
 *
*
 *       OSBYTE  10   Duration of second colour
*
 *
*

*************************************************************************

 ;on entry Y=0 or 1 if from FX 9 call, X=new value


 E6B2    LDA     &0252,Y ;get mark period count
 E6B5    PHA             ;push it
 E6B6    TXA             ;get new count
 E6B7    STA     &0252,Y ;store it
 E6BA    PLA             ;get back original value
 E6BB    TAY             ;put it in Y
 E6BC    LDA     &0251   ;get value of flash counter
 E6BF    BNE     &E6D1   ;if not zero E6D1

 E6C1    STX     &0251   ;else restore old value
 E6C4    LDA     &0248   ;get current video ULA control register setting
 E6C7    PHP             ;push flags
 E6C8    ROR             ;rotate bit 0 into carry, carry into bit 7
 E6C9    PLP             ;get back flags
 E6CA    ROL             ;rotate back carry into bit 0
 E6CB    STA     &0248   ;store it in RAM copy
 E6CE    STA     &FE20   ;and ULA control register

 E6D1    BVC     &E6AD   ;then exit via OSBYTE 7/8



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

```
 *
*
 *       OSBYTE  2   select input stream
*
 *
*

*********************************************************************

 ;on input X contains stream number

 E6D3    TXA             ;A=X
 E6D4    AND     #&01    ;blank out bits 1 - 7
 E6D6    PHA             ;push A
 E6D7    LDA     &0250   ;and get current ACIA control setting
 E6DA    ROL             ;Bit 7 into carry
 E6DB    CPX     #&01    ;if X>=1 then
 E6DD    ROR             ;bit 7 of A=1
 E6DE    CMP     &0250   ;compare this with ACIA control setting
 E6E1    PHP             ;push processor
 E6E2    STA     &0250   ;put A into ACIA control setting
 E6E5    STA     &FE08   ;and write to control register
 E6E8    JSR     &E173   ;set up RS423 buffer
 E6EB    PLP             ;get back P
 E6EC    BEQ     &E6F1   ;if new setting different from old E6F1 else
 E6EE    BIT     &FE09   ;set bit 6 and 7

 E6F1    LDX     &0241   ;get current input buffer number
 E6F4    PLA             ;get back A
 E6F5    STA     &0241   ;store it
 E6F8    RTS             ;and return



*********************************************************************
 *
*
 *       OSBYTE  13   disable events
*
 *
*

*********************************************************************

         ;X contains event number 0-9

 E6F9    TYA             ;Y=0 A=0



*********************************************************************
 *
*
 *       OSBYTE  14   enable events
*
 *
*
```

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

         ;X contains event number 0-9


 E6FA    CPX     #&0A    ;if X>9
 E6FC    BCS     &E6AE   ;goto E6AE for exit
 E6FE    LDY     &02BF,X ;else get event enable flag
 E701    STA     &02BF,X ;store new value in flag
 E704    BVC     &E6AD   ;and exit via E6AD



*************************************************************************
 *
*
 *       OSBYTE  16   Select A/D channel
*
 *
*

*************************************************************************
  ;X contains channel number or 0 if disable conversion

 E706    BEQ     &E70B   ;if X=0 then E70B
 E708    JSR     &DE8C   ;start conversion

 E70B    LDA     &024D   ;get  current maximum ADC channel number
 E70E    STX     &024D   ;store new value
 E711    TAX             ;put old value in X
 E712    RTS             ;and exit
                         ;



*************************************************************************
 *
*
 *       OSBYTE 129   Read key within time limit
*
 *
*

*************************************************************************
  ;X and Y contains either time limit in centi seconds Y=&7F max
  ; or Y=&FF and X=-ve INKEY value

 E313    TYA             ;A=Y
 E714    BMI     &E721   ;if Y=&FF the E721
 E716    CLI             ;else allow interrupts
 E717    JSR     &DEBB   ;and go to timed routine
 E71A    BCS     &E71F   ;if carry set then E71F
 E71C    TAX             ;then X=A
 E71D    LDA     #&00    ;A=0

 E71F    TAY             ;Y=A
 E720    RTS             ;and return
                         ;
```

```
                        ;scan keyboard
 E721    TXA            ;A=X
 E722    EOR     #&7F   ;convert to keyboard input
 E724    TAX            ;X=A
 E725    JSR     &F068  ;then scan keyboard
 E728    ROL            ;put bit 7 into carry
 E729    LDX     #&FF   ;X=&FF
 E72B    LDY     #&FF   ;Y=&FF
 E72D    BCS     &E731  ;if bit 7 of A was set goto E731 (RTS)
 E72F    INX            ;else X=0
 E730    INY            ;and Y=0
 E731    RTS            ;and exit


 ********** check occupancy of input or free space of output buffer
*******
                        ;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

 E732    TXA            ;buffer number in A
 E733    EOR     #&FF   ;invert it
 E735    TAX            ;X=A
 E736    CPX     #&02   ;is X>1
 E738    CLV            ;clear V flag
 E739    BVC     &E73E  ;and goto E73E count buffer

 E73B    BIT     &D9B7  ;set V
 E73E    JMP     (&022E) ;CNPV defaults to E1D1


 ************* check RS423 input buffer
**********************************

 E741    SEC
 E742    LDX     #&01   ;X=1 to point to buffer
 E744    JSR     &E738  ;and count it
 E747    CPY     #&01   ;if the hi byte of the answer is 1 or more
 E749    BCS     &E74E  ;then Return
 E74B    CPX     &025B  ;else compare with minimum buffer space
 E74E    RTS            ;and exit




*************************************************************************
 *
*
 *       OSBYTE 128  READ ADC CHANNEL
*
```

```
 *
*

*********************************************************************

 ;ON Entry: X=0               Exit Y contains number of last channel
converted
 ;          X=channel number       X,Y contain 16 bit value read from
channe
 ;          X<0 Y=&FF              X returns information about various
buffers
 ;          X=&FF (keyboard )      X=number of characters in buffer
 ;          X=&FE (RS423 Input)    X=number of characters in buffer
 ;          X=&FD (RS423 output)   X=number of empty spaces in buffer
 ;          X=&FC (Printer)        X=number of empty spaces in buffer
 ;          X=&FB (sound 0)        X=number of empty spaces in buffer
 ;          X=&FA (sound 1)        X=number of empty spaces in buffer
 ;          X=&F9 (sound 2)        X=number of empty spaces in buffer
 ;          X=&F8 (sound 3)        X=number of empty spaces in buffer
 ;          X=&F7 (Speech )        X=number of empty spaces in buffer



 E74F    BMI     &E732   ;if X is -ve then E732 count spaces
 E751    BEQ     &E75F   ;if X=0 then E75F
 E753    CPX     #&05    ;else check for Valid channel
 E755    BCS     &E729   ;if not E729 set X & Y to 0 and exit
 E757    LDY     &02B9,X ;get conversion values for channel of interest
Hi &
 E75A    LDA     &02B5,X ;lo byte
 E75D    TAX             ;X=lo byte
 E75E    RTS             ;and exit


 E75F    LDA     &FE40   ;read system VIA port B
 E762    ROR             ;move high nybble to low
 E763    ROR             ;
 E764    ROR             ;
 E765    ROR             ;
 E766    EOR     #&FF    ;and invert it
 E768    AND     #&03    ;isolate the FIRE buttons
 E76A    LDY     &02BE   ;get analogue system flag byte
 E76D    STX     &02BE   ;store X here
 E770    TAX             ;A=X bits 0 and 1 indicate fire buttons
 E771    RTS             ;and return

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

```
*********************************************************************
*
 **
**
 **      OSBYTE  DEFAULT ENTRY POINT
**
 **
**
 **      pointed to by default BYTEV
**
 **
**

*********************************************************************
*

*********************************************************************
*


 E772    PHA             ;save A
 E773    PHP             ;save Processor flags
 E774    SEI             ;disable interrupts
 E775    STA     &EF     ;store A,X,Y in zero page
 E777    STX     &F0     ;
 E779    STY     &F1     ;
 E77B    LDX     #&07    ;X=7 to signal osbyte is being attempted
 E77D    CMP     #&75    ;if A=0-116
 E77F    BCC     &E7C2   ;then E7C2
 E781    CMP     #&A1    ;if A<161
 E783    BCC     &E78E   ;then E78E
 E785    CMP     #&A6    ;if A=161-165
 E787    BCC     &E7C8   ;then EC78
 E789    CLC             ;clear carry

 E78A    LDA     #&A1    ;A=&A1
 E78C    ADC     #&00    ;

 ********* process osbyte calls 117 - 160 ****************************

 E78E    SEC             ;set carry
 E78F    SBC     #&5F    ;convert to &16 to &41 (22-65)

 E791    ASL             ;double it (44-130)
 E792    SEC             ;set carry


 E793    STY     &F1     ;store Y
 E795    TAY             ;Y=A
 E796    BIT     &025E   ;read econet intercept flag
 E799    BPL     &E7A2   ;if no econet intercept required E7A2

 E79B    TXA             ;else A=X
 E79C    CLV             ;V=0
 E79D    JSR     &E57E   ; to JMP via ECONET vector
 E7A0    BVS     &E7BC   ;if return with V set E7BC


 E7A2    LDA     &E5B4,Y ;get address from table
```

```
 E7A5    STA     &FB     ;store it as hi byte
 E7A7    LDA     &E5B3,Y ;repeat for lo byte
 E7AA    STA     &FA     ;
 E7AC    LDA     &EF     ;restore A
 E7AE    LDY     &F1     ;Y
 E7B0    BCS     &E7B6   ;if carry is set E7B6

 E7B2    LDY     #&00    ;else
 E7B4    LDA     (&F0),Y ;get value from address pointed to by &F0/1
(Y,X)

 E7B6    SEC             ;set carry
 E7B7    LDX     &F0     ;restore X
 E7B9    JSR     &F058   ;call &FA/B

 E7BC    ROR             ;C=bit 0
 E7BD    PLP             ;get back flags
 E7BE    ROL             ;bit 0=Carry
 E7BF    PLA             ;get back A
 E7C0    CLV             ;clear V
 E7C1    RTS             ;and exit


 *************** Process OSBYTE CALLS BELOW &75
**************************

 E7C2    LDY     #&00    ;Y=0
 E7C4    CMP     #&16    ;if A<&16
 E7C6    BCC     &E791   ;goto E791

 E7C8    PHP             ;push flags
 E7C9    PHP             ;push flags

 E7CA    PLA             ;pull flags
 E7CB    PLA             ;pull flags
 E7CC    JSR     &F168   ;offer paged ROMS service 7/8 unrecognised
osbyte/word
 E7CF    BNE     &E7D6   ;if roms don't recognise it then E7D6
 E7D1    LDX     &F0     ;else restore X
 E7D3    JMP     &E7BC   ;and exit


 E7D6    PLP             ;else pull flags
 E7D7    PLA             ;and A
 E7D8    BIT     &D9B7   ;set V and C
 E7DB    RTS             ;and return

 E7DC    LDA     &EB     ;read cassette critical flag bit 7 = busy
 E7DE    BMI     &E812   ;if busy then EB12


 E7E0    LDA     #&08    ;else A=8 to check current Catalogue status
 E7E2    AND     &E2     ;by anding with CFS status flag
 E7E4    BNE     &E7EA   ;if not set (not in use) then E7EA RTS
 E7E6    LDA     #&88    ;A=%10001000
 E7E8    AND     &BB     ;AND with FS options (short msg bits)
 E7EA    RTS             ;RETURN
```

```
*********************************************************************
*
*********************************************************************
*
 **
**
 **      OSWORD  DEFAULT ENTRY POINT
**
 **
**
 **      pointed to by default WORDV
**
 **
**
*********************************************************************
*
*********************************************************************
*


 E7EB    PHA             ;Push A
 E7EC    PHP             ;Push flags
 E7ED    SEI             ;disable interrupts
 E7EE    STA     &EF     ;store A,X,Y
 E7F0    STX     &F0     ;
 E7F2    STY     &F1     ;
 E7F4    LDX     #&08    ;X=8
 E7F6    CMP     #&E0    ;if A=>224
 E7F8    BCS     &E78A   ;then E78A with carry set

 E7FA    CMP     #&0E    ;else if A=>14
 E7FC    BCS     &E7C8   ;else E7C8 with carry set pass to ROMS & exit

 E7FE    ADC     #&44    ;add to form pointer to table
 E800    ASL             ;double it
 E801    BCC     &E793   ;goto E793 ALWAYS!! (carry clear E7F8)
                         ;this reads bytes from table and enters routine




*********************************************************************
 *
*
 *       OSWORD  05   ENTRY POINT
*
 *
*
 *       read a byte from I/O memory
*
 *
*
```

```
*********************************************************************
 ;block of 4 bytes set at address pointed to by 00F0/1 (Y,X)
 ;XY +0  ADDRESS of byte
 ;   +4  on exit byte read

 E803    JSR     &E815   ;set up address of data block
 E806    LDA     (&F9,X) ;get byte
 E808    STA     (&F0),Y ;store it
 E80A    RTS             ;exit




*********************************************************************
 *
*
 *       OSWORD  06   ENTRY POINT
*
 *
*
 *       write a byte to I/O memory
*
 *
*
*********************************************************************
 ;block of 5 bytes set at address pointed to by 00F0/1 (Y,X)
 ;XY +0  ADDRESS of byte
 ;   +4  byte to be written


 E80B    JSR     &E815   ;set up address
 E80E    LDA     (&F0),Y ;get byte
 E810    STA     (&F9,X) ;store it
 E812    LDA     #&00    ;a=0
 E814    RTS             ;exit


 ********************: set up data block
*********************************

 E815    STA     &FA     ;&FA=A
 E817    INY             ;Y=1
 E818    LDA     (&F0),Y ;get byte from block
 E81A    STA     &FB     ;store it
 E81C    LDY     #&04    ;Y=4
 E81E    LDX     #&01    ;X=1
 E820    RTS             ;and exit




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

```
 *       OSBYTE  00   ENTRY POINT
*
 *
*
 *       read OS version number
*
 *
*

*********************************************************************


 E821    BNE     &E81E   ;if A <> 0 then exit else print error
 E823    BRK             ;
 E824    DB      &F7             ;error number
 E825    DB      'OS 1.20'       ;error message
 E82C    BRK




*********************************************************************
 *
*
 *       OSWORD  07   ENTRY POINT
*
 *
*
 *       make a sound
*
 *
*

*********************************************************************
 ;block of 8 bytes set at address pointed to by 00F0/1
 ;XY +0  Channel or +0=Flush, Channel:+1=Hold,Sync
 ;    2  Amplitude
 ;    4  Pitch
 ;    6  Duration
 ;Y=0 on entry

 E82D    INY             ;increment Y
 E82E    LDA     (&F0),Y ;get channel byte from table
 E830    CMP     #&FF    ;if its &FF goto speech
 E832    BEQ     &E88D   ;at E8DD
 E834    CMP     #&20    ;else if>=&20 set carry
 E836    LDX     #&08    ;X=8 for unrecognised OSWORD call
 E838    BCS     &E7CA   ;and if carry set off to E7CA to offer to ROMS
 E83A    DEY             ;Y=0
 E83B    JSR     &E8C9   ;return with Carry set if Flush=1:A=C
 E83E    ORA     #&04    ;convert to buffer number
 E840    TAX             ;A=X
 E841    BCC     &E848   ;and if carry clear (ex E8C9) then E848

 E843    JSR     &E1AE   ;else flush buffer
 E846    LDY     #&01    ;Y=1

 E848    JSR     &E8C9   ;returns with carry set if H=1;A=S
```

```
 E84B    STA     &FA     ;Sync number =&FA
 E84D    PHP             ;ave flags
 E84E    LDY     #&06    ;Y=6
 E850    LDA     (&F0),Y ;Get Duration byte
 E852    PHA             ;push it

 E853    LDY     #&04    ;Y=4
 E855    LDA     (&F0),Y ;get pitch byte
 E857    PHA             ;push it
 E858    LDY     #&02    ;get amplitude byte
 E85A    LDA     (&F0),Y ;
 E85C    ROL             ;H now =bit 0 (carry shifted)
 E85D    SEC             ;set carry
 E85E    SBC     #&02    ;subract 2
 E860    ASL             ;multiply by 4
 E861    ASL             ;
 E862    ORA     &FA     ;add S byte (0-3)

                         ;at this point bit 7=0 for an envelope
                         ;bits 3-6 give envelope -1, or volume +15
                         ;(i.e.complemented)
                         ;bit 2 gives H
                         ;bits 0-1 =Sync
 E864    JSR     &E1F8   ;transfer to buffer
 E867    BCC     &E887   ;if C set on exit succesful transfer so E887

 E869    PLA             ;else get back
 E86A    PLA             ;stored values
 E86B    PLP             ;
```

```
*************************************************************************
 *
*
 *       OSBYTE  117   ENTRY POINT
*
 *
*
 *       read VDU status
*
 *
*
*************************************************************************
```

```
 E86C    LDX     &D0     ;get VDU status byte in X
 E86E    RTS             ;and return
```

```
 ************* set up sound data for Bell
*******************************
```

```
 E86F    PHP             ;push P
 E870    SEI             ;bar interrupts
```

```
 E871    LDA     &0263   ;get bell channel number in A
 E874    AND     #&07    ; (bits 0-3 only set)
 E876    ORA     #&04    ;set bit 2
 E878    TAX             ;X=A = bell channel number +4=buffer number
 E879    LDA     &0264   ;get bell amplitude/envelope number
 E87C    JSR     &E4B0   ;store it in buffer pointed to by X
 E87F    LDA     &0266   ;get bell duration
 E882    PHA             ;save it
 E883    LDA     &0265   ;get bell frequency
 E886    PHA             ;save it
 E887    SEC             ;set carry

 E888    ROR     &0800,X ;and pass into bit 7 to warn that channel is
active
 E88B    BMI     &E8A4   ;goto E8A4 ALWAYS!!


 **********:speech handling routine
************************************

 E88D    PHP             ;push flags
 E88E    INY             ;Y=2
 E88F    LDA     (&F0),Y ;get byte
 E891    PHA             ;store it
 E892    INY             ;Y=4
 E893    LDA     (&F0),Y ;get byte
 E895    PHA             ;store it
 E896    LDY     #&00    ;Y=0
 E898    LDA     (&F0),Y ;get byte
 E89A    LDX     #&08    ;X=8
 E89C    JSR     &E1F8   ;select speech buffer and pass A
 E89F    BCS     &E869   ;if carry set on return restore stack
                         ;and exit
 E8A1    ROR     &02D7   ;else clear bit 7 of buffer busy flag to show
                         ;buffer has been set
 E8A4    PLA             ;get back byte
 E8A5    JSR     &E4B0   ;enter it in buffer X
 E8A8    PLA             ;get back next
 E8A9    JSR     &E4B0   ;and enter it again
 E8AC    PLP             ;get back flags
 E8AD    RTS             ;and exit



*************************************************************************
 *
*
 *       OSWORD  08   ENTRY POINT
*
 *
*
 *       define envelope
*
 *
*

*************************************************************************
 ;block of 14 bytes set at address pointed to by 00F0/1
 ;XY +0  Envelope number, also in A
```

```
 ;    1  bits 0-6 length of each step in centi-secsonds bit 7=0 auto
repeat
 ;    2  change of Pitch per step (-128-+127) in section 1
 ;    3  change of Pitch per step (-128-+127) in section 2
 ;    4  change of Pitch per step (-128-+127) in section 3
 ;    5  number of steps in section 1 (0-255)
 ;    6  number of steps in section 2 (0-255)
 ;    7  number of steps in section 3 (0-255)
 ;    8  change of amplitude per step during attack  phase (-127 to +127)
 ;    9  change of amplitude per step during decay   phase (-127 to +127)
 ;   10  change of amplitude per step during sustain phase (-127 to +127)
 ;   11  change of amplitude per step during release phase (-127 to +127)
 ;   12  target level at end of attack phase   (0-126)
 ;   13  target level at end of decay  phase   (0-126)

 ;Y=0 on entry


 E8AE    SBC     #&01    ;set up appropriate displacement to storage area
 E8B0    ASL             ;A=(A-1)*16 or 15
 E8B1    ASL             ;
 E8B2    ASL             ;
 E8B3    ASL             ;
 E8B4    ORA     #&0F    ;
 E8B6    TAX             ;X=A
 E8B7    LDA     #&00    ;A=0

 E8B9    LDY     #&10    ;Y=10

 E8BB    CPY     #&0E    ;is Y>=14??
 E8BD    BCS     &E8C1   ;yes then E8C1
 E8BF    LDA     (&F0),Y ;else get byte from parameter block
 E8C1    STA     &08C0,X ;and store it in appropriate area
 E8C4    DEX             ;decrement X
 E8C5    DEY             ;Decrement Y
 E8C6    BNE     &E8BB   ;if not 0 then do it again
 E8C8    RTS             ;else exit
                         ;note that envelope number is NOT transferred

 E8C9    LDA     (&F0),Y ;get byte
 E8CB    CMP     #&10    ;is it greater than 15, if so set carry
 E8CD    AND     #&03    ;and 3 to clear bits 2-7
 E8CF    INY             ;increment Y
 E8D0    RTS             ;and exit




*************************************************************************
 *
*
 *       OSWORD  03   ENTRY POINT
*
 *
*
 *       read interval timer
*
 *
*
```

```
*************************************************************************
 F0/1 points to block to store data

 E8D1    LDX     #&0F    ;X=&F displacement from clock to timer
 E8D3    BNE     &E8D8   ;jump to E8D8




*************************************************************************
 *
*
 *       OSWORD  01   ENTRY POINT
*
 *
*
 *       read system clock
*
 *
*

*************************************************************************
 F0/1 points to block to store data


 E8D5    LDX     &0283   ;X=current system clock store pointer

 E8D8    LDY     #&04    ;Y=4
 E8DA    LDA     &028D,X ;read byte
 E8DD    STA     (&F0),Y ;store it in parameter block
 E8DF    INX             ;X=x+1
 E8E0    DEY             ;Y=Y-1
 E8E1    BPL     &E8DA   ;if Y>0 then do it again
 E8E3    RTS             ;else exit




*************************************************************************
 *
*
 *       OSWORD  04   ENTRY POINT
*
 *
*
 *       write interval timer
*
 *
*

*************************************************************************
 F0/1 points to block to store data


 E8E4    LDA     #&0F    ;offset between clock and timer
 E8E6    BNE     &E8EE   ;jump to E8EE ALWAYS!!
```

```
*************************************************************************
 *
*
 *       OSWORD  02   ENTRY POINT
*
 *
*
 *       write system clock
*
 *
*

*************************************************************************
 F0/1 points to block to store data


 E8E8    LDA     &0283   ;get current clock store pointer
 E8EB    EOR     #&0F    ;and invert to get inactive timer
 E8ED    CLC             ;clear carry

 E8EE    PHA             ;store A
 E8EF    TAX             ;X=A
 E8F0    LDY     #&04    ;Y=4
 E8F2    LDA     (&F0),Y ;and transfer all 5 bytes
 E8F4    STA     &028D,X ;to the clock or timer
 E8F7    INX             ;
 E8F8    DEY             ;
 E8F9    BPL     &E8F2   ;if Y>0 then E8F2
 E8FB    PLA             ;get back stack
 E8FC    BCS     &E8E3   ;if set (write to timer) E8E3 exit
 E8FE    STA     &0283   ;write back current clock store
 E901    RTS             ;and exit




*************************************************************************
 *
*
 *       OSWORD  00   ENTRY POINT
*
 *
*
 *       read line from current input to memory
*
 *
*

*************************************************************************
 ;F0/1 points to parameter block
 ;       +0/1    buffer address for input
 ;       +2      Maximum line length
 ;       +3      minimum acceptable ASCII value
 ;       +4      maximum acceptable ASCII value
```

```
 E902    LDY     #&04    ;Y=4

 E904    LDA     (&F0),Y ;transfer bytes 4,3,2 to 2B3-2B5
 E906    STA     &02B1,Y ;
 E909    DEY             ;decrement Y
 E90A    CPY     #&02    ;until Y=1
 E90C    BCS     &E904   ;

 E90E    LDA     (&F0),Y ;get address of input buffer
 E910    STA     &E9     ;store it in &E9 as temporary buffer
 E912    DEY             ;decrement Y
 E913    STY     &0269   ;Y=0 store in print line counter for paged mode
 E916    LDA     (&F0),Y ;get lo byte of address
 E918    STA     &E8     ;and store in &E8
 E91A    CLI             ;allow interrupts
 E91B    BCC     &E924   ;Jump to E924

 E91D    LDA     #&07    ;A=7
 E91F    DEY             ;decrement Y
 E920    INY             ;increment Y
 E921    JSR     OSWRCH  ;and call OSWRCH

 E924    JSR     OSRDCH  ;else read character  from input stream
 E927    BCS     &E972   ;if carry set then illegal character or other
error
                         ;exit via E972
 E929    TAX             ;X=A
 E92A    LDA     &027C   ;A=&27C get character destination status
 E92D    ROR             ;put VDU driver bit in carry
 E92E    ROR             ;if this is 1 VDU driver is disabled
 E92F    TXA             ;X=A
 E930    BCS     &E937   ;if Carry set E937
 E932    LDX     &026A   ;get number of items in VDU queque
 E935    BNE     &E921   ;if not 0 output character and loop round again

 E937    CMP     #&7F    ;if character is not delete
 E939    BNE     &E942   ;goto E942
 E93B    CPY     #&00    ;else is Y=0
 E93D    BEQ     &E924   ;and goto E924
 E93F    DEY             ;decrement Y
 E940    BCS     &E921   ;and if carry set E921 to output it
 E942    CMP     #&15    ;is it delete line &21
 E944    BNE     &E953   ;if not E953
 E946    TYA             ;else Y=A, if its 0 we are still reading first
                         ;character
 E947    BEQ     &E924   ;so E924
 E949    LDA     #&7F    ;else output DELETES

 E94B    JSR     OSWRCH  ;until Y=0
 E94E    DEY             ;
 E94F    BNE     &E94B   ;

 E951    BEQ     &E924   ;then read character again

 E953    STA     (&E8),Y ;store character in designated buffer
 E955    CMP     #&0D    ;is it CR?
 E957    BEQ     &E96C   ;if so E96C
 E959    CPY     &02B3   ;else check the line length
 E95C    BCS     &E91D   ;if = or greater loop to ring bell
```

```
 E95E    CMP     &02B4   ;check minimum character
 E961    BCC     &E91F   ;if less than minimum backspace
 E963    CMP     &02B5   ;check maximum character
 E966    BEQ     &E920   ;if equal E920
 E968    BCC     &E920   ;or less E920
 E96A    BCS     &E91F   ;then JUMP E91F

 E96C    JSR     OSNEWL  ;output CR/LF
 E96F    JSR     &E57E   ;call Econet vector

 E972    LDA     &FF     ;A=ESCAPE FLAG
 E974    ROL             ;put bit 7 into carry
 E975    RTS             ;and exit routine




*************************************************************************
 *
*
 *       OSBYTE  05   ENTRY POINT
*
 *
*
 *       SELECT PRINTER TYPE
*
 *
*
*************************************************************************


 E976    CLI             ;allow interrupts briefly
 E977    SEI             ;bar interrupts
 E978    BIT     &FF     ;check if ESCAPE is pending
 E97A    BMI     &E9AC   ;if it is E9AC
 E97C    BIT     &02D2   ;else check bit 7 buffer 3 (printer)
 E97F    BPL     &E976   ;if not empty bit 7=0 E976

 E981    JSR     &E1A4   ;check for user defined routine
 E984    LDY     #&00    ;Y=0
 E986    STY     &F1     ;F1=0




*************************************************************************
 *
*
 *       OSBYTE  01   ENTRY POINT
*
 *
*
 *       READ/WRITE USER FLAG (&281)
*
```

```
 *
*
 *          AND
*
 *
*
 *       OSBYTE  06   ENTRY POINT
*
 *
*
 *       SET PRINTER IGNORE CHARACTER
*
 *
*

*********************************************************************
 ; A contains osbyte number

 E988    ORA     #&F0    ;A=osbyte +&F0
 E98A    BNE     &E99A   ;JUMP to E99A




*********************************************************************
 *
*
 *       OSBYTE  0C   ENTRY POINT
*
 *
*
 *       SET  KEYBOARD AUTOREPEAT RATE
*
 *
*

*********************************************************************


 E98C    BNE     &E995   ;if &F0<>0 goto E995
 E98E    LDX     #&32    ;if X=0 in original call then X=32
 E990    STX     &0254   ;to set keyboard autorepeat delay ram copy
 E993    LDX     #&08    ;X=8




*********************************************************************
 *
*
 *       OSBYTE  0B   ENTRY POINT
*
 *
*
 *       SET  KEYBOARD AUTOREPEAT DELAY
*
```

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

 E995    ADC     #&CF    ;A=A+&D0 (carry set)



*************************************************************************
 *
*
 *       OSBYTE  03   ENTRY POINT
*
 *
*
 *       SELECT OUTPUT STREAM
*
 *
*
 *               AND
*
 *
*
 *
*
 *       OSBYTE  04   ENTRY POINT
*
 *
*
 *       ENABLE/DISABLE CURSOR EDITING
*
 *
*
*************************************************************************

 E997    CLC             ;c,ear carry
 E998    ADC     #&E9    ;A=A+&E9

 E99A    STX     &F0     ;sto2e X



*************************************************************************
 *
*
 *       OSBYTE  A6-FF   ENTRY POINT
*
 *
*
```

```
 *       READ/ WRITE SYSTEM VARIABLE OSBYTE NO. +&190
*
 *
*
```

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

```
 E99C    TAY             ;Y=A
 E99D    LDA     &0190,Y ;i.e. A=&190 +osbyte call!
 E9A0    TAX             ;preserve this
 E9A1    AND     &F1     ;new value = OLD value AND Y EOR X!
 E9A3    EOR     &F0     ;
 E9A5    STA     &0190,Y ;store it
 E9A8    LDA     &0191,Y ;get value of next byte into A
 E9AB    TAY             ;Y=A
 E9AC    RTS             ;and exit
```

```
 ******* SERIAL BAUD RATE LOOK UP TABLE
**********************************
```

```
 E9AD    DB      &64     ; % 01100100      75
 E9AE    DB      &7F     ; % 01111111     150
 E9AF    DB      &5B     ; % 01011011     300
 E9B0    DB      &6D     ; % 01101101    1200
 E9B1    DB      &C9     ; % 11001001    2400
 E9B2    DB      &F6     ; % 11110110    4800
 E9B3    DB      &D2     ; % 11010010    9600
 E9B4    DB      &E4     ; % 11100100   19200
 E9B5    DB      &40     ; % 01000000
```

```
**************************************************************************
 *
*
 *       OSBYTE  &13   ENTRY POINT
*
 *
*
 *       Wait for Animation
*
 *
*
```

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

```
 E9B6    LDA     &0240   ;read vertical sync counter
 E9B9    CLI             ;allow interrupts briefly
 E9BA    SEI             ;bar interrupts
 E9BB    CMP     &0240   ;is it 0 (Vertical sync pulse)
 E9BE    BEQ     &E9B9   ;no then E9B9
```

```
*********************************************************************
 *
*
 *       OSBYTE  160   ENTRY POINT
*
 *
*
 *       READ VDU VARIABLE
*
 *
*

*********************************************************************
 ;X contains the variable number
 ;0 =lefthand column in pixels current graphics window
 ;2 =Bottom row in pixels current graphics window
 ;4 =Right hand column in pixels current graphics window
 ;6 =Top row in pixels current graphics window
 ;8 =lefthand column in absolute characters current text window
 ;9 =Bottom row in absolute characters current text window
 ;10 =Right hand column i. absolute characters current text window
 ;11 =Top row in absolute characters current text window
 ;12-15 current graphics origin in external coordinates
 ;16-19 current graphics cursor in external coordina4es
 ;20-23 old graphics cursor in internal coordinates
 ;24 current text cursor in X and Y

 E9C0    LDY     &0301,X ;get VDU variable hi
 E9C3    LDA     &0300,X ;low
 E9C6    TAX             ;X=low byte
 E9C7    RTS             ;and exit




*********************************************************************
 *
*
 *       OSBYTE  18   ENTRY POINT
*
 *
*
 *       RESET SOFT KEYS
*
 *
*

*********************************************************************

 E9C8    LDA     #&10    ;set consistency flag
 E9CA    STA     &0284   ;

 E9CD    LDX     #&00    ;X=0

 E9CF    STA     &0B00,X ;and wipe clean
 E9D2    INX             ;soft key buffer
 E9D3    BNE     &E9CF   ;until X=0 again
```

```
 E9D5    STX     &0284   ;zero consistency flag
 E9D8    RTS             ;and exit




*************************************************************************
 *
*
 *        OSBYTE &76 (118) SET LEDs to Keyboard Status
*
 *
*

*************************************************************************


                         ;osbyte entry with carry set
                         ;called from &CB0E, &CAE3, &DB8B

 E9D9    PHP             ;PUSH P
 E9DA    SEI             ;DISABLE INTERUPTS
 E9DB    LDA     #&40    ;switch on CAPS and SHIFT lock lights
 E9DD    JSR     &E9EA   ;via subroutine
 E9E0    BMI     &E9E7   ;if ESCAPE exists (M set) E9E7
 E9E2    CLC             ;else clear V and C
 E9E3    CLV             ;before calling main keyboard routine to
 E9E4    JSR     &F068   ;switch on lights as required
 E9E7    PLP             ;get back flags
 E9E8    ROL             ;and rotate carry into bit 0
 E9E9    RTS             ;Return to calling routine
                         ;
 ***************** Turn on keyboard lights and Test Escape flag
************
                         ;called from &E1FE, &E9DD
                         ;
 E9EA    BCC     &E9F5   ;if carry clear
 E9EC    LDY     #&07    ;switch on shift lock light
 E9EE    STY     &FE40   ;
 E9F1    DEY             ;Y=6
 E9F2    STY     &FE40   ;switch on Caps lock light
 E9F5    BIT     &FF     ;set minus flag if bit 7 of &00FF is set
indicating
 E9F7    RTS             ;that ESCAPE condition exists, then return
                         ;
 ****************** Write A to SYSTEM VIA register B
************************
                         ;called from &CB6D, &CB73
 E9F8    PHP             ;push flags
 E9F9    SEI             ;disable interupts
 E9FA    STA     &FE40   ;write register B from Accumulator
 E9FD    PLP             ;get back flags
 E9FE    RTS             ;and exit
                         ;
```

```
*************************************************************************
 *
*
 *       OSBYTE 154 (&9A) SET VIDEO ULA
*
 *
*

*************************************************************************


 E9FF    TXA             ;osbyte entry! X transferred to A thence to

 *******Set Video ULA control register **entry from VDU routines
**************
                         ;called from &CBA6, &DD37

 EA00    PHP             ;save flags
 EA01    SEI             ;disable interupts
 EA02    STA     &0248   ;save RAM copy of new parameter
 EA05    STA     &FE20   ;write to control register
 EA08    LDA     &0253   ;read  space count
 EA0B    STA     &0251   ;set flash counter to this value
 EA0E    PLP             ;get back status
 EA0F    RTS             ;and return




*************************************************************************
 *
*
 *        OSBYTE &9B (155) write to pallette register
*
 *
*

*************************************************************************

                ;entry X contains value to write
 EA10    TXA             ;A=X
 EA11    EOR     #&07    ;convert to palette format
 EA13    PHP             ;
 EA14    SEI             ;prevent interrupts
 EA15    STA     &0249   ;store as current palette setting
 EA18    STA     &FE21   ;store actual colour in register
 EA1B    PLP             ;get back flags
 EA1C    RTS             ;and exit
                         ;



*************************************************************************
 *       GSINIT  string initialisation
*
 *       F2/3 points to string offset by Y
*
```

```
 *
*
 *       ON EXIT
*
 *       Z flag set indicates null string,
*
 *       Y points to first non blank character
*
 *       A contains first non blank character
*

*************************************************************************

 EA1D    CLC             ;clear carry

 EA1E    ROR     &E4     ;Rotate moves carry to &E4
 EA20    JSR     &E03A   ;get character from text
 EA23    INY             ;increment Y to point at next character
 EA24    CMP     #&22    ;check to see if its '"'
 EA26    BEQ     &EA2A   ;if so EA2A (carry set)
 EA28    DEY             ;decrement Y
 EA29    CLC             ;clear carry
 EA2A    ROR     &E4     ;move bit 7 to bit 6 and put carry in bit 7
 EA2C    CMP     #&0D    ;check to see if its CR to set Z
 EA2E    RTS             ;and return



*************************************************************************
 *       GSREAD  string read routine
*
 *       F2/3 points to string offset by Y
*
 *
*

*************************************************************************
                         ;
 EA2F    LDA     #&00    ;A=0
 EA31    STA     &E5     ;store A
 EA33    LDA     (&F2),Y ;read first character
 EA35    CMP     #&0D    ;is it CR
 EA37    BNE     &EA3F   ;if not goto EA3F
 EA39    BIT     &E4     ;if bit 7=1 no 2nd '"' found
 EA3B    BMI     &EA8F   ;goto EA8F
 EA3D    BPL     &EA5A   ;if not EA5A

 EA3F    CMP     #&20    ;is less than a space?
 EA41    BCC     &EA8F   ;goto EA8F
 EA43    BNE     &EA4B   ;if its not a space EA4B
 EA45    BIT     &E4     ;is bit 7 of &E4 =1
 EA47    BMI     &EA89   ;if so goto EA89
 EA49    BVC     &EA5A   ;if bit 6 = 0 EA5A
 EA4B    CMP     #&22    ;is it '"'
 EA4D    BNE     &EA5F   ;if not EA5F
 EA4F    BIT     &E4     ;if so and Bit 7 of &E4 =0 (no previous ")
 EA51    BPL     &EA89   ;then EA89
 EA53    INY             ;else point at next character
 EA54    LDA     (&F2),Y ;get it
 EA56    CMP     #&22    ;is it '"'
```

```
EA58    BEQ     &EA89   ;if so then EA89

EA5A    JSR     &E03A   ;read a byte from text
EA5D    SEC             ;and return with
EA5E    RTS             ;carry set
                        ;
EA5F    CMP     #&7C    ;is it '|'
EA61    BNE     &EA89   ;if not EA89
EA63    INY             ;if so increase Y to point to next character
EA64    LDA     (&F2),Y ;get it
EA66    CMP     #&7C    ;and compare it with '|' again
EA68    BEQ     &EA89   ;if its '|' then EA89
EA6A    CMP     #&22    ;else is it '"'
EA6C    BEQ     &EA89   ;if so then EA89
EA6E    CMP     #&21    ;is it !
EA70    BNE     &EA77   ;if not then EA77
EA72    INY             ;increment Y again
EA73    LDA     #&80    ;set bit 7
EA75    BNE     &EA31   ;loop back to EA31 to set bit 7 in next CHR
EA77    CMP     #&20    ;is it a space
EA79    BCC     &EA8F   ;if less than EA8F Bad String Error
EA7B    CMP     #&3F    ;is it '?'
EA7D    BEQ     &EA87   ;if so EA87
EA7F    JSR     &EABF   ;else modify code as if CTRL had been pressed
EA82    BIT     &D9B7   ;if bit 6 set
EA85    BVS     &EA8A   ;then EA8A
EA87    LDA     #&7F    ;else set bits 0 to 6 in A

EA89    CLV             ;clear V
EA8A    INY             ;increment Y
EA8B    ORA     &E5     ;
EA8D    CLC             ;clear carry
EA8E    RTS             ;Return
                        ;
EA8F    BRK             ;
EA90    DB      &FD     ;error number
EA93    DB 'Bad String' ; message
EA9B    BRK             ;


 ************ Modify code as if SHIFT pressed
*****************************

EA9C    CMP     #&30    ;if A='0' skip routine
EA9E    BEQ     &EABE   ;
EAA0    CMP     #&40    ;if A='@' skip routine
EAA2    BEQ     &EABE   ;
EAA4    BCC     &EAB8   ;if A<'@' then EAB8
EAA6    CMP     #&7F    ;else is it DELETE




EAA8    BEQ     &EABE   ;if so skip routine
EAAA    BCS     &EABC   ;if greater than &7F then toggle bit 4
EAAC    EOR     #&30    ;reverse bits 4 and 5
EAAE    CMP     #&6F    ;is it &6F (previously ' _' (&5F))
EAB0    BEQ     &EAB6   ;goto EAB6
```

```
EAB2    CMP     #&50    ;is it &50 (previously '`' (&60))
EAB4    BNE     &EAB8   ;if not EAB8
EAB6    EOR     #&1F    ;else continue to convert ` _
EAB8    CMP     #&21    ;compare &21 '!'
EABA    BCC     &EABE   ;if less than return
EABC    EOR     #&10    ;else finish conversion by toggling bit 4
EABE    RTS             ;exit
                        ;
                        ;ASCII codes &00 &20 no change
                        ;21-3F have bit 4 reverses (31-3F)
                        ;41-5E A-Z have bit 5 reversed a-z
                        ;5F & 60 are reversed
                        ;61-7E bit 5 reversed a-z becomes A-Z
                        ;DELETE unchanged
                        ;&80+ has bit 4 changed


 ************** Implement CTRL codes
************************************


EABF    CMP     #&7F    ;is it DEL
EAC1    BEQ     &EAD1   ;if so ignore routine
EAC3    BCS     &EAAC   ;if greater than &7F go to EAAC
EAC5    CMP     #&60    ;if A<>'`'
EAC7    BNE     &EACB   ;goto EACB
EAC9    LDA     #&5F    ;if A=&60, A=&5F

EACB    CMP     #&40    ;if A<&40
EACD    BCC     &EAD1   ;goto EAD1  and return unchanged
EACF    AND     #&1F    ;else zero bits 5 to 7
EAD1    RTS             ;return
                        ;
EAD2    DB      '/!BOOT'
EAD8    DB      &0D
```

OS SERIES 8
GEOFF COX

```
*********************************************************************
 *
*
 *        OSBYTE &F7 (247) INTERCEPT BREAK
*
 *
*
*********************************************************************


 EAD9    LDA     &0287   ;get BREAK vector code
 EADC    EOR     #&4C    ;produces 0 if JMP not in &287
 EADE    BNE     &EAF3   ;if not goto EAF3
 EAE0    JMP     &0287   ;else jump to user BREAK code




*********************************************************************
 *
*
 *        OSBYTE &90 (144)   *TV
*
 *
*
*********************************************************************

                ;X=display delay
                ;Y=interlace flag

 EAE3    LDA     &0290   ;VDU vertical adjustment
 EAE6    STX     &0290   ;store new value
 EAE9    TAX             ;put old value in X
 EAEA    TYA             ;put interlace flag in A
 EAEB    AND     #&01    ;maximum value =1
 EAED    LDY     &0291   ;get old value into Y
 EAF0    STA     &0291   ;put new value into A
 EAF3    RTS             ;and Exit
                         ;




*********************************************************************
 *
*
 *        OSBYTE &93 (147)  WRITE TO FRED
*
 *
*
*********************************************************************
```

```
                        ;X is offset within page
                        ;Y is byte to write
 EAF4    TYA            ;
 EAF5    STA     &FC00,X ;
 EAF8    RTS            ;




*********************************************************************
 *
*
 *        OSBYTE &95 (149)  WRITE TO JIM
*
 *
*
*********************************************************************

                        ;X is offset within page
                        ;Y is byte to write
                        ;
 EAF9    TYA            ;
 EAFA    STA     &FD00,X ;
 EAFD    RTS            ;
                        ;




*********************************************************************
 *
*
 *        OSBYTE &97 (151)  WRITE TO SHEILA
*
 *
*
*********************************************************************

                        ;X is offset within page
                        ;Y is byte to write

 EAFE    TYA            ;
 EAFF    STA     &FE00,X ;
 EB02    RTS            ;
                        ;


 ****************** Silence a sound channel
*******************************
                        ;X=channel number

 EB03    LDA     #&04    ;mark end of release phase
 EB05    STA     &0808,X ;to channel X
 EB08    LDA     #&C0    ;load code for zero volume
```

```
 ****** if sound not disabled set sound generator volume
******************

 EB0A    STA     &0804,X ;store A to give basic sound level of Zero
 EB0D    LDY     &0262   ;get sound output/enable flag
 EB10    BEQ     &EB14   ;if sound enabled goto EB14
 EB12    LDA     #&C0    ;else load zero sound code
 EB14    SEC             ;set carry
 EB15    SBC     #&40    ;subtract &40
 EB17    LSR             ;divide by 8
 EB18    LSR             ;to get into bits 0 - 3
 EB19    LSR             ;
 EB1A    EOR     #&0F    ;invert bits 0-3
 EB1C    ORA     &EB3C,X ;get channel number into top nybble
 EB1F    ORA     #&10    ;


 EB21    PHP             ;

 EB22    SEI             ;disable interrupts
 EB23    LDY     #&FF    ;System VIA port A all outputs
 EB25    STY     &FE43   ;set
 EB28    STA     &FE4F   ;output A on port A
 EB2B    INY             ;Y=0
 EB2C    STY     &FE40   ;enable sound chip
 EB2F    LDY     #&02    ;set and
 EB31    DEY             ;execute short delay
 EB32    BNE     &EB31   ;
 EB34    LDY     #&08    ;then disable sound chip again
 EB36    STY     &FE40   ;
 EB39    LDY     #&04    ;set delay
 EB3B    DEY             ;and loop delay
 EB3C    BNE     &EB3B   ;
 EB3E    PLP             ;get back flags
 EB3F    RTS             ;and exit

 *******: Sound parameters look up table
*********************************

 EB40    DB      &E0,&C0,&A0,&80


 EB44    JMP     &EC59   ;just to allow relative branches in early part
                         ;of sound interrupt routine



*************************************************************************
 *
*
 *       PROCESS SOUND INTERRUPT
*

*************************************************************************

 EB47    LDA     #&00    ;
 EB49    STA     &083B   ;zero number of channels on hold for sync
 EB4C    LDA     &0838   ;get number of channels required for sync
 EB4F    BNE     &EB57   ;if this <>0 then EB57
 EB51    INC     &083B   ;else number of chanels on hold for sync =1
```

```
 EB54    DEC     &0838   ;number of channels required for sync =255

 EB57    LDX     #&08    ;set loop counter
 EB59    DEX             ;loop
 EB5A    LDA     &0800,X ;get value of &800 +offset (sound queue
occupancy)
 EB5D    BEQ     &EB44   ;if 0 goto EC59 no sound this channel
 EB5F    LDA     &02CF,X ;else get buffer busy flag
 EB62    BMI     &EB69   ;if negative (buffer empty) goto EB69
 EB64    LDA     &0818,X ;else if duration count not zer0
 EB67    BNE     &EB6C   ;goto EB6C
 EB69    JSR     &EC6B   ;check and pick up new sound if required
 EB6C    LDA     &0818,X ;if duration count 0
 EB6F    BEQ     &EB84   ;goto EB84
 EB71    CMP     #&FF    ;else if it is &FF (infinite duration)
 EB73    BEQ     &EB87   ;go onto EB87
 EB75    DEC     &081C,X ;decrement 10 mS count
 EB78    BNE     &EB87   ;and if 0
 EB7A    LDA     #&05    ;reset to 5
 EB7C    STA     &081C,X ;to give 50 mSec delay
 EB7F    DEC     &0818,X ;and decrement main counter
 EB82    BNE     &EB87   ;if not zero then EB87
 EB84    JSR     &EC6B   ;else check and get nw sound
 EB87    LDA     &0824,X ;if step progress counter is 0 no envelope
involved
 EB8A    BEQ     &EB91   ;so jump to EB91
 EB8C    DEC     &0824,X ;else decrement it
 EB8F    BNE     &EB44   ;and if not zero go on to EC59
 EB91    LDY     &0820,X ;get  envelope data offset from (8C0)
 EB94    CPY     #&FF    ;if 255 no envelope set so
 EB96    BEQ     &EB44   ;goto EC59
 EB98    LDA     &08C0,Y ;else get get step length
 EB9B    AND     #&7F    ;zero repeat bit
 EB9D    STA     &0824,X ;and store it
 EBA0    LDA     &0808,X ;get phase counter
 EBA3    CMP     #&04    ;if release phase completed
 EBA5    BEQ     &EC07   ;goto EC07
 EBA7    LDA     &0808,X ;else start new step by getting phase
 EBAA    CLC             ;
 EBAB    ADC     &0820,X ;add it to interval multiplier
 EBAE    TAY             ;transfer to Y
 EBAF    LDA     &08CB,Y ;and get target value base for envelope
 EBB2    SEC             ;
 EBB3    SBC     #&3F    ;
 EBB5    STA     &083A   ;store modified number as current target
amplitude
 EBB8    LDA     &08C7,Y ;get byte from envelope store
 EBBB    STA     &0839   ;store as current amplitude step
 EBBE    LDA     &0804,X ;get base volumelevel
 EBC1    PHA             ;save it
 EBC2    CLC             ;clear carry
 EBC3    ADC     &0839   ;add to current amplitude step
 EBC6    BVC     &EBCF   ;if no overflow
 EBC8    ROL             ;double it Carry = bit 7
 EBC9    LDA     #&3F    ;if bit =1 A=&3F
 EBCB    BCS     &EBCF   ;into &EBCF
 EBCD    EOR     #&FF    ;else toggle bits (A=&C0)

                         ;at this point the BASIC volume commands are
converted
```

```
                        ; &C0 (0) to &38 (-15) 3 times, In fact last 3
bits
                        ;are ignored so &3F represents -15

 EBCF    STA     &0804,X ;store in current volume
 EBD2    ROL             ;multiply by 2
 EBD3    EOR     &0804,X ;if bits 6 and 7 are equal
 EBD6    BPL     &EBE1   ;goto &EBE1
 EBD8    LDA     #&3F    ;if carry clear A=&3F (maximum)
 EBDA    BCC     &EBDE   ;or
 EBDC    EOR     #&FF    ;&C0 minimum

 EBDE    STA     &0804,X ;and this is stored in current volume

 EBE1    DEC     &0839   ;decrement amplitude change per step
 EBE4    LDA     &0804,X ;get volume again
 EBE7    SEC             ;set carry
 EBE8    SBC     &083A   ;subtract target value
 EBEB    EOR     &0839   ;negative value undicates correct trend
 EBEE    BMI     &EBF9   ;so jump to next part
 EBF0    LDA     &083A   ;else enter new phase
 EBF3    STA     &0804,X ;
 EBF6    INC     &0808,X ;

 EBF9    PLA             ;get the old volume level
 EBFA    EOR     &0804,X ;and compare with the old
 EBFD    AND     #&F8    ;
 EBFF    BEQ     &EC07   ;if they are the same goto EC07
 EC01    LDA     &0804,X ;else set new level
 EC04    JSR     &EB0A   ;via EB0A
 EC07    LDA     &0810,X ;get absolute pitch value
 EC0A    CMP     #&03    ;if it =3
 EC0C    BEQ     &EC59   ;skip rest of loop as all sections are finished
 EC0E    LDA     &0814,X ;else if 814,X is not 0 current section is not
                         ;complete
 EC11    BNE     &EC3D   ;so EC3D
 EC13    INC     &0810,X ;else implement a section change
 EC16    LDA     &0810,X ;check if its complete
 EC19    CMP     #&03    ;if not
 EC1B    BNE     &EC2D   ;goto EC2D
 EC1D    LDY     &0820,X ;else set A from
 EC20    LDA     &08C0,Y ;&820 and &8C0 (first envelope byte)
 EC23    BMI     &EC59   ;if negative there is no repeat
 EC25    LDA     #&00    ;else restart section sequence
 EC27    STA     &0830,X ;
 EC2A    STA     &0810,X ;

 EC2D    LDA     &0810,X ;get number of steps in new section
 EC30    CLC             ;
 EC31    ADC     &0820,X ;
 EC34    TAY             ;
 EC35    LDA     &08C4,Y ;
 EC38    STA     &0814,X ;set in 814+X
 EC3B    BEQ     &EC59   ;and if 0 then EC59

 EC3D    DEC     &0814,X ;decrement
 EC40    LDA     &0820,X ;and pick up rate of pitch change
 EC43    CLC             ;
 EC44    ADC     &0810,X ;
 EC47    TAY             ;
```

```
EC48    LDA     &08C1,Y ;
EC4B    CLC             ;
EC4C    ADC     &0830,X ;add to rate of differential pitch change
EC4F    STA     &0830,X ;and save it
EC52    CLC             ;
EC53    ADC     &080C,X ;ad to base pitch
EC56    JSR     &ED01   ;and set new pitch

EC59    CPX     #&04    ;if X=4 (last channel)
EC5B    BEQ     &EC6A   ;goto EC6A (RTS)
EC5D    JMP     &EB59   ;else do loop again

EC60    LDX     #&08    ;X=7 again
EC62    DEX             ;loop
EC63    JSR     &ECA2   ;clear channel
EC66    CPX     #&04    ;if not 4
EC68    BNE     &EC62   ;do it again
EC6A    RTS             ;and return
                        ;
EC6B    LDA     &0808,X ;check for last channel
EC6E    CMP     #&04    ;is it 4 (release complete)
EC70    BEQ     &EC77   ;if so EC77
EC72    LDA     #&03    ;else mark release in progress
EC74    STA     &0808,X ;and store it
EC77    LDA     &02CF,X ;is buffer not empty
EC7A    BEQ     &EC90   ;if so EC90
EC7C    LDA     #&00    ;else mark buffer not empty
EC7E    STA     &02CF,X ;an store it

EC81    LDY     #&04    ;loop counter
EC83    STA     &082B,Y ;zero sync bytes
EC86    DEY             ;
EC87    BNE     &EC83   ;until Y=0

EC89    STA     &0818,X ;zero duration count
EC8C    DEY             ;and set sync count to
EC8D    STY     &0838   ;&FF
EC90    LDA     &0828,X ;get synchronising flag
EC93    BEQ     &ECDB   ;if its 0 then ECDB
EC95    LDA     &083B   ;else get number of channels on hold
EC98    BEQ     &ECD0   ;if 0 then ECD0
EC9A    LDA     #&00    ;else
EC9C    STA     &0828,X ;zero note length interval
EC9F    JMP     &ED98   ;and goto ED98

ECA2    JSR     &EB03   ;silence the channel
ECA5    TYA             ;Y=0 A=Y
ECA6    STA     &0818,X ;zero main count
ECA9    STA     &02CF,X ;mark buffer not empty
ECAC    STA     &0800,X ;mark channel dormant
ECAF    LDY     #&03    ;loop counter
ECB1    STA     &082C,Y ;zero sync flags
ECB4    DEY             ;
ECB5    BPL     &ECB1   ;

ECB7    STY     &0838   ;number of channels to &FF
ECBA    BMI     &ED06   ;jump to ED06 ALWAYS

ECBC    PHP             ;save flags
ECBD    SEI             ;and disable interrupts
```

```
 ECBE    LDA     &0808,X ;check for end of release
 ECC1    CMP     #&04    ;
 ECC3    BNE     &ECCF   ;and if not found ECCF
 ECC5    JSR     &E45B   ;elseexamine buffer
 ECC8    BCC     &ECCF   ;if not empty ECCF
 ECCA    LDA     #&00    ;else mark channel dormant
 ECCC    STA     &0800,X ;
 ECCF    PLP             ;get back flags

 ECD0    LDY     &0820,X ;if no envelope 820=&FF
 ECD3    CPY     #&FF    ;
 ECD5    BNE     &ECDA   ;then terminate sound
 ECD7    JSR     &EB03   ;via EB03
 ECDA    RTS             ;else return
                         ;


 ************ Synchronise sound routines
**********************************

 ECDB    JSR     &E45B   ;examine buffer if empty carry set
 ECDE    BCS     &ECBC   ;
 ECE0    AND     #&03    ;else examine next word if>3 or 0
 ECE2    BEQ     &EC9F   ;goto ED98 (via EC9F)
 ECE4    LDA     &0838   ;else get synchronising count
 ECE7    BEQ     &ECFE   ;in 0 (complete) goto ECFE
 ECE9    INC     &0828,X ;else set sync flag
 ECEC    BIT     &0838   ;if 0838 is +ve S has already been set so
 ECEF    BPL     &ECFB   ;jump to ECFB
 ECF1    JSR     &E45B   ;else get first byte
 ECF4    AND     #&03    ;mask bits 0,1
 ECF6    STA     &0838   ;and store result
 ECF9    BPL     &ECFE   ;Jump to ECFE (ALWAYS!!)

 ECFB    DEC     &0838   ;decrement 0838
 ECFE    JMP     &ECD0   ;and silence the channel if envelope not in use


 ************ Pitch setting
*********************************************

 ED01    CMP     &082C,X ;If A=&82C,X then pitch is unchanged
 ED04    BEQ     &ECDA   ;then exit via ECDA
 ED06    STA     &082C,X ;store new pitch
 ED09    CPX     #&04    ;if X<>4 then not noise so
 ED0B    BNE     &ED16   ;jump to ED16


 *********** Noise setting
***********************************************

 ED0D    AND     #&0F    ;convert to chip format
 ED0F    ORA     &EB3C,X ;
 ED12    PHP             ;save flags
 ED13    JMP     &ED95   ;and pass to chip control routine at EB22 via
ED95


 ED16    PHA             ;
 ED17    AND     #&03    ;
```

```
ED19    STA     &083C   ;lose eigth tone surplus
ED1C    LDA     #&00    ;
ED1E    STA     &083D   ;
ED21    PLA             ;get back A
ED22    LSR             ;divide by 12
ED23    LSR             ;
ED24    CMP     #&0C    ;
ED26    BCC     &ED2F   ;
ED28    INC     &083D   ;store result
ED2B    SBC     #&0C    ;with remainder in A
ED2D    BNE     &ED24   ;

                        ;at this point 83D defines the Octave
                        ;A the semitone within the octave
ED2F    TAY             ;Y=A
ED30    LDA     &083D   ;get octave number into A
ED33    PHA             ;push it
ED34    LDA     &EDFB,Y ;get byte from look up table
ED37    STA     &083D   ;store it
ED3A    LDA     &EE07,Y ;get byte from second table
ED3D    PHA             ;push it
ED3E    AND     #&03    ;keep two LS bits only
ED40    STA     &083E   ;save them
ED43    PLA             ;pull second table byte
ED44    LSR             ;push hi nybble into lo nybble
ED45    LSR             ;
ED46    LSR             ;
ED47    LSR             ;
ED48    STA     &083F   ;store it
ED4B    LDA     &083D   ;get back octave number
ED4E    LDY     &083C   ;adjust for surplus eighth tones
ED51    BEQ     &ED5F   ;
ED53    SEC             ;
ED54    SBC     &083F   ;
ED57    BCS     &ED5C   ;
ED59    DEC     &083E   ;
ED5C    DEY             ;
ED5D    BNE     &ED53   ;
ED5F    STA     &083D   ;
ED62    PLA             ;
ED63    TAY             ;
ED64    BEQ     &ED6F   ;
ED66    LSR     &083E   ;
ED69    ROR     &083D   ;
ED6C    DEY             ;
ED6D    BNE     &ED66   ;
ED6F    LDA     &083D   ;
ED72    CLC             ;
ED73    ADC     &C43D,X ;
ED76    STA     &083D   ;
ED79    BCC     &ED7E   ;
ED7B    INC     &083E   ;
ED7E    AND     #&0F    ;
ED80    ORA     &EB3C,X ;
ED83    PHP             ;push P
ED84    SEI             ;bar interrupts
ED85    JSR     &EB21   ;set up chip access 1
ED88    LDA     &083D   ;
ED8B    LSR     &083E   ;
ED8E    ROR             ;
```

```
 ED8F    LSR     &083E   ;
 ED92    ROR             ;
 ED93    LSR             ;
 ED94    LSR             ;
 ED95    JMP     &EB22   ;set up chip access 2 and return


 ***************** Pick up and interpret sound buffer data
*****************

 ED98    PHP             ;push flags
 ED99    SEI             ;disable interrupts
 ED9A    JSR     &E460   ;read a byte from buffer
 ED9D    PHA             ;push A
 ED9E    AND     #&04    ;isolate H bit
 EDA0    BEQ     &EDB7   ;if 0 then EDB7
 EDA2    PLA             ;get back A
 EDA3    LDY     &0820,X ;if &820,X=&FF
 EDA6    CPY     #&FF    ;envelope is not in use
 EDA8    BNE     &EDAD   ;
 EDAA    JSR     &EB03   ;so call EB03 to silence channel

 EDAD    JSR     &E460   ;clear buffer of redundant data
 EDB0    JSR     &E460   ;and again
 EDB3    PLP             ;get back flags
 EDB4    JMP     &EDF7   ;set main duration count using last byte from
buffer

 EDB7    PLA             ;get back A
 EDB8    AND     #&F8    ;zero bits 0-2
 EDBA    ASL             ;put bit 7 into carry
 EDBB    BCC     &EDC8   ;if zero (envelope) jump to EDC8
 EDBD    EOR     #&FF    ;invert A
 EDBF    LSR             ;shift right
 EDC0    SEC             ;
 EDC1    SBC     #&40    ;subtract &40
 EDC3    JSR     &EB0A   ;and set volume
 EDC6    LDA     #&FF    ;A=&FF

 EDC8    STA     &0820,X ;get envelope no.-1 *16 into A
 EDCB    LDA     #&05    ;set duration sub-counter
 EDCD    STA     &081C,X ;
 EDD0    LDA     #&01    ;set phase counter
 EDD2    STA     &0824,X ;
 EDD5    LDA     #&00    ;set step counter
 EDD7    STA     &0814,X ;
 EDDA    STA     &0808,X ;and envelope phase
 EDDD    STA     &0830,X ;and pitch differential
 EDE0    LDA     #&FF    ;
 EDE2    STA     &0810,X ;set step count
 EDE5    JSR     &E460   ;read pitch
 EDE8    STA     &080C,X ;set it
 EDEB    JSR     &E460   ;read buffer
 EDEE    PLP             ;
 EDEF    PHA             ;save duration
 EDF0    LDA     &080C,X ;get back pitch value
 EDF3    JSR     &ED01   ;and set it
 EDF6    PLA             ;get back duration
 EDF7    STA     &0818,X ;set it
```

```
EDFA    RTS             ;and return


********** Pitch look up table 1*****************************
EDFB        DB          &F0
EDFC        DB          &B7
EDFD        DB          &82
EDFE        DB          &4F
EDFF        DB          &20
EE00        DB          &F3
EE01        DB          &C8
EE02        DB          &A0
EE03        DB          &7B
EE04        DB          &57
EE05        DB          &35
EE06        DB          &16


********** Pitch look up table 2
*****************************

EE07        DB          &E7
EE08        DB          &D7
EE09        DB          &CB
EE0A        DB          &C3
EE0B        DB          &B7
EE0C        DB          &AA
EE0D        DB          &A2
EE0E        DB          &9A
EE0F        DB          &92
EE10        DB          &8A
EE11        DB          &82
EE12        DB          &7A

*********: set current filing system ROM/PHROM
*************************
EE13    LDA     #&EF    ;get ROM
EE15    STA     &F5     ;store it
EE17    RTS             ;return



********** Get byte from data ROM
***************************************

EE18    LDX     #&0D    ;X=13
EE1A    INC     &F5     ;
EE1C    LDY     &F5     ;get Rom
EE1E    BPL     &EE59   ;if +ve its a sideways ROM else its a PHROM
EE20    LDX     #&00    ;PHROM
EE22    STX     &F7     ;set address pointer in PHROM
EE24    INX             ;
EE25    STX     &F6     ;to 0001
EE27    JSR     &EEBB   ;pass info to speech processor
EE2A    LDX     #&03    ;X=3

EE2C    JSR     &EE62   ;check for speech processor and output until
                        ;it reports, read byte from ROM
EE2F    CMP     &DF0C,X ;if A<> DF0C+X then EE18 (DF0C = (C))
EE32    BNE     &EE18   ;
```

```
 EE34    DEX             ;else decrement X
 EE35    BPL     &EE2C   ;and do it again
 EE37    LDA     #&3E    ;
 EE39    STA     &F6     ;get noe lo byte address
 EE3B    JSR     &EEBB   ;pass info to speech processor
 EE3E    LDX     #&FF    ;
 EE40    JSR     &EE62   ;check for speech proc. etc.
 EE43    LDY     #&08    ;
 EE45    ASL             ;
 EE46    ROR     &F7,X   ;
 EE48    DEY             ;
 EE49    BNE     &EE45   ;
 EE4B    INX             ;
 EE4C    BEQ     &EE40   ;
 EE4E    CLC             ;
 EE4F    BCC     &EEBB   ;


 ************ ROM SERVICE
*************************************************

 EE51    LDX     #&0E    ;
 EE53    LDY     &F5     ;if Y is negative (PHROM)
 EE55    BMI     &EE62   ;GOTO EE62
 EE57    LDY     #&FF    ;else Y=255
 EE59    PHP             ;push flags
 EE5A    JSR     &F168   ;offer paged rom service
 EE5D    PLP             ;pull processor flags
 EE5E    CMP     #&01    ;if A>0  set carry
 EE60    TYA             ;A=Y
 EE61    RTS             ;return

 ********* PHROM SERVICE
*************************************************
                         ;
 EE62    PHP             ;push processor flags
 EE63    SEI             ;disable interrupts
 EE64    LDY     #&10    ;Y=16
 EE66    JSR     &EE7F   ;call EE7F (osbyte 159 write to speech processor
 EE69    LDY     #&00    ;Y=0
 EE6B    BEQ     &EE84   ;Jump to EE84 (ALWAYS!!)



*************************************************************************
 *
*
 *       OSBYTE 158 read from speech processor *
 *
*

*************************************************************************


 EE6D    LDY     #&00    ;Y=0 to set speech proc to read
 EE6F    BEQ     &EE82   ;jump to EE82 always

                         ;write A to speech processor as two nybbles
```

```
 EE71    PHA             ;push A
 EE72    JSR     &EE7A   ;to write to speech processor
 EE75    PLA             ;get back A
 EE76    ROR             ;bring upper nybble to lower nybble
 EE77    ROR             ;by rotate right
 EE78    ROR             ;4 times
 EE79    ROR             ;

 EE7A    AND     #&0F    ;Y=lo nybble A +&40
 EE7C    ORA     #&40    ;
 EE7E    TAY             ;forming command for speech processor



*************************************************************************
 *
*
 *       OSBYTE 159 Write to speech processor
*
 *
*

*************************************************************************


 ;       on entry data or command in Y

 EE7F    TYA             ;transfer command to A
 EE80    LDY     #&01    ;to set speech proc to write

                         ;if Y=0 read speech processor
                         ;if Y=1 write speech processor

 EE82    PHP             ;push flags
 EE83    SEI             ;disable interrupts
 EE84    BIT     &027B   ;test for prescence of speech processor
 EE87    BPL     &EEAA   ;if not there goto EEAA
 EE89    PHA             ;else push A
 EE8A    LDA     &F075,Y ;
 EE8D    STA     &FE43   ;set DDRA of system VIA to give 8 bit input
(Y=0)
                         ;or 8 bit output (Y=1)
 EE90    PLA             ;get back A
 EE91    STA     &FE4F   ;and send to speech chip
 EE94    LDA     &F077,Y ;output Prt B of system VIA
 EE97    STA     &FE40   ;to select read or write (dependent on Y)
 EE9A    BIT     &FE40   ;loop until
 EE9D    BMI     &EE9A   ;speech proceessor reports ready (bit 7 Prt B=0)
 EE9F    LDA     &FE4F   ;read speech processor data if  input selected
 EEA2    PHA             ;push A
 EEA3    LDA     &F079,Y ;reset speech
 EEA6    STA     &FE40   ;processor
 EEA9    PLA             ;get back A

 EEAA    PLP             ;get back flags
 EEAB    TAY             ;transfer A to Y
 EEAC    RTS             ;and exit routine
                         ;
```

```
EEAD    LDA     &03CB   ;set rom displacement pointer
EEB0    STA     &F6     ;in &F6
EEB2    LDA     &03CC   ;
EEB5    STA     &F7     ;And &F7
EEB7    LDA     &F5     ;if F5 is +ve ROM is selected so
EEB9    BPL     &EED9   ;goto EED9

EEBB    PHP             ;else push processor
EEBC    SEI             ;disable interrupts
EEBD    LDA     &F6     ;get lo displacement
EEBF    JSR     &EE71   ;pass two nyblles to speech proc.
EEC2    LDA     &F5     ;&FA=&F5
EEC4    STA     &FA     ;
EEC6    LDA     &F7     ;get hi displacement value
EEC8    ROL             ;replace two most significant bits of A
EEC9    ROL             ;by 2 LSBs of &FA
EECA    LSR     &FA     ;
EECC    ROR             ;
EECD    LSR     &FA     ;
EECF    ROR             ;
EED0    JSR     &EE71   ;pass two nybbles to speech processor
EED3    LDA     &FA     ;FA has now been divided by 4 so pass
EED5    JSR     &EE7A   ;lower nybble to speech proc.
EED8    PLP             ;get back flags
EED9    RTS             ;and Return
                        ;



************ Keyboard Input and housekeeping ************************
        ;entered from &F00C

EEDA    LDX     #&FF    ;
EEDC    LDA     &EC     ;get value of most recently pressed key
EEDE    ORA     &ED     ;Or it with previous key to check for presses
EEE0    BNE     &EEE8   ;if A=0 no keys pressed so off you go
EEE2    LDA     #&81    ;else enable keybd interupt only by writing bit
7
EEE4    STA     &FE4E   ;and bit 0 of system VIA interupt register
EEE7    INX             ;set X=0
EEE8    STX     &0242   ;reset keyboard semaphore


**********: Turn on Keyboard indicators ******************************
EEEB    PHP             ;save flags
EEEC    LDA     &025A   ;read keyboard status;
                        ;Bit 7  =1 shift enabled
                        ;Bit 6  =1 control pressed
                        ;bit 5  =0 shift lock
                        ;Bit 4  =0 Caps lock
                        ;Bit 3  =1 shift pressed
EEEF    LSR             ;shift Caps bit into bit 3
EEF0    AND     #&18    ;mask out all but 4 and 3
EEF2    ORA     #&06    ;returns 6 if caps lock OFF &E if on
EEF4    STA     &FE40   ;turn on or off caps light if required
EEF7    LSR             ;bring shift bit into bit 3
EEF8    ORA     #&07    ;
EEFA    STA     &FE40   ;turn on or off shift  lock light
EEFD    JSR     &F12E   ;set keyboard counter
EF00    PLA             ;get back flags
```

```
EF01    RTS             ;return
                        ;


*********************************************************************
 *
*
 * MAIN KEYBOARD HANDLING ROUTINE   ENTRY FROM KEYV
*
 * ==========================================================
*
 *
*
 *                       ENTRY CONDITIONS
*
 *                       ================
*
 * C=0, V=0 Test Shift and CTRL keys.. exit with N set if CTRL pressed
*
 *                                  ........with V set if Shift pressed
*
 *
*
 * C=1, V=0 Scan Keyboard as OSBYTE &79
*
 *
*
 * C=0, V=1 Key pressed interrupt entry
*
 *
*
 * C=1, V=1 Timer interrupt entry
*
 *
*
 *
*

*********************************************************************

EF02    BVC     &EF0E   ;if V is clear then leave interrupt routine
EF04    LDA     #&01    ;disable keyboard interrupts
EF06    STA     &FE4E   ;by writing to VIA interrupt vector
EF09    BCS     &EF13   ;if timer interrupt then EF13
EF0B    JMP     &F00F   ;else to F00F

EF0E    BCC     &EF16   ;if test SHFT & CTRL goto EF16
EF10    JMP     &F0D1   ;else to F0D1
                        ;to scan keyboard

*********************************************************************
 *       Timer interrupt entry
*

*********************************************************************

EF13    INC     &0242   ;increment keyboard semaphore (to 0)
```

```
*************************************************************************
 *       Test Shift and Control Keys entry
*
*************************************************************************

 EF16    LDA     &025A   ;read keyboard status;
                         ;Bit 7  =1 shift enabled
                         ;Bit 6  =1 control pressed
                         ;bit 5  =0 shift lock
                         ;Bit 4  =0 Caps lock
                         ;Bit 3  =1 shift pressed
 EF19    AND     #&B7    ;zero bits 3 and 6
 EF1B    LDX     #&00    ;zero X to test for shift key press
 EF1D    JSR     &F02A   ;interrogate keyboard X=&80 if key determined by
                         ;X on entry is pressed
 EF20    STX     &FA     ;save X
 EF22    CLV             ;clear V
 EF23    BPL     &EF2A   ;if no key press (X=0) then EF2A else
 EF25    BIT     &D9B7   ;set M and V
 EF28    ORA     #&08    ;set bit 3 to indicate Shift was pressed
 EF2A    INX             ;check the control key
 EF2B    JSR     &F02A   ;via keyboard interrogate
 EF2E    BCC     &EEEB   ;if carry clear (entry via EF16) then off to
EEEB
                         ;to turn on keyboard lights as required
 EF30    BPL     &EF34   ;if key not pressed goto EF30
 EF32    ORA     #&40    ;or set CTRL pressed bit in keyboard status byte
in A
 EF34    STA     &025A   ;save status byte
 EF37    LDX     &EC     ;if no key previously pressed
 EF39    BEQ     &EF4D   ;then EF4D
 EF3B    JSR     &F02A   ;else check to see if key still pressed
 EF3E    BMI     &EF50   ;if so enter repeat routine at EF50
 EF40    CPX     &EC     ;else compare X with last key pressed (set
flags)
 EF42    STX     &EC     ;store X in last key pressed
 EF44    BNE     &EF4D   ;if different from previous (Z clear) then EF4D
 EF46    LDX     #&00    ;else zero
 EF48    STX     &EC     ;last key pressed
 EF4A    JSR     &F01F   ;and reset repeat system
 EF4D    JMP     &EFE9   ;


 ********** REPEAT ACTION
*********************************************
 EF50    CPX     &EC     ;if X<>than last key pressed
 EF52    BNE     &EF42   ;then back to EF42
 EF54    LDA     &E7     ;else get value of AUTO REPEAT COUNTDOWN TIMER
 EF56    BEQ     &EF7B   ;if 0 goto EF7B
 EF58    DEC     &E7     ;else decrement
 EF5A    BNE     &EF7B   ;and if not 0 goto EF7B
                         ;this means that either the repeat system is
dormant
                         ;or it is not at the end of its count
 EF5C    LDA     &02CA   ;next value for countdown timer
 EF5F    STA     &E7     ;store it
 EF61    LDA     &0255   ;get auto repeat rate from 0255
 EF64    STA     &02CA   ;store it as next value for Countdown timer
```

```
 EF67    LDA     &025A   ;get keyboard status
 EF6A    LDX     &EC     ;get last key pressed
 EF6C    CPX     #&D0    ;if not SHIFT LOCK key (&D0) goto
 EF6E    BNE     &EF7E   ;EF7E
 EF70    ORA     #&90    ;sets shift enabled, & no caps lock all else
preserved
 EF72    EOR     #&A0    ;reverses shift lock disables Caps lock and
Shift enab
 EF74    STA     &025A   ;reset keyboard status
 EF77    LDA     #&00    ;and set timer
 EF79    STA     &E7     ;to 0
 EF7B    JMP     &EFE9   ;

 EF7E    CPX     #&C0    ;if not CAPS LOCK
 EF80    BNE     &EF91   ;goto EF91
 EF82    ORA     #&A0    ;sets shift enabled and disables SHIFT LOCK
 EF84    BIT     &FA     ;if bit 7 not set by (EF20) shift NOT pressed
 EF86    BPL     &EF8C   ;goto EF8C
 EF88    ORA     #&10    ;else set CAPS LOCK not enabled
 EF8A    EOR     #&80    ;reverse SHIFT enabled

 EF8C    EOR     #&90    ;reverse both SHIFT enabled and CAPs Lock
 EF8E    JMP     &EF74   ;reset keyboard status and set timer


 *********** get ASCII code ********************************************
         ;on entry X=key pressed internal number

 EF91    LDA     &EFAB,X ;get code from look up table
 EF94    BNE     &EF99   ;if not zero goto EF99 else TAB pressed
 EF96    LDA     &026B   ;get TAB character

 EF99    LDX     &025A   ;get keyboard status
 EF9C    STX     &FA     ;store it in &FA
 EF9E    ROL     &FA     ;rotate to get CTRL pressed into bit 7
 EFA0    BPL     &EFA9   ;if CTRL NOT pressed EFA9

 EFA2    LDX     &ED     ;get no. of previously pressed key
 EFA4    BNE     &EF4A   ;if not 0 goto EF4A to reset repeat system etc.
 EFA6    JSR     &EABF   ;else perform code changes for CTRL

 EFA9    ROL     &FA     ;move shift lock into bit 7
 EFAB    BMI     &EFB5   ;if not effective goto EFB5 else
 EFAD    JSR     &EA9C   ;make code changes for SHIFT

 EFB0    ROL     &FA     ;move CAPS LOCK into bit 7
 EFB2    JMP     &EFC1   ;and Jump to EFC1

 EFB5    ROL     &FA     ;move CAPS LOCK into bit 7
 EFB7    BMI     &EFC6   ;if not effective goto EFC6
 EFB9    JSR     &E4E3   ;else make changes for CAPS LOCK on, return with
                         ;C clear for Alphabetic codes
 EFBC    BCS     &EFC6   ;if carry set goto EFC6 else make changes for
 EFBE    JSR     &EA9C   ;SHIFT as above

 EFC1    LDX     &025A   ;if shift enabled bit is clear
 EFC4    BPL     &EFD1   ;goto EFD1
 EFC6    ROL     &FA     ;else get shift bit into 7
 EFC8    BPL     &EFD1   ;if not set goto EFD1
 EFC8    BPL     &EFD1   ;if not set goto EFD1
```

```
 EFCA    LDX     &ED     ;get previous key press
 EFCC    BNE     &EFA4   ;if not 0 reset repeat system etc. via EFA4
 EFCE    JSR     &EA9C   ;else make code changes for SHIFT
 EFD1    CMP     &026C   ;if A<> ESCAPE code
 EFD4    BNE     &EFDD   ;goto EFDD
 EFD6    LDX     &0275   ;get Escape key status
 EFD9    BNE     &EFDD   ;if ESCAPE returns ASCII code goto EFDD
 EFDB    STX     &E7     ;store in Auto repeat countdown timer

 EFDD    TAY             ;
 EFDE    JSR     &F129   ;disable keyboard
 EFE1    LDA     &0259   ;read Keyboard disable flag used by Econet
 EFE4    BNE     &EFE9   ;if keyboard locked goto EFE9
 EFE6    JSR     &E4F1   ;put character in input buffer
 EFE9    LDX     &ED     ;get previous keypress
 EFEB    BEQ     &EFF8   ;if none  EFF8
 EFED    JSR     &F02A   ;examine to see if key still pressed
 EFF0    STX     &ED     ;store result
 EFF2    BMI     &EFF8   ;if pressed goto EFF8
 EFF4    LDX     #&00    ;else zero X
 EFF6    STX     &ED     ;and &ED

 EFF8    LDX     &ED     ;get &ED
 EFFA    BNE     &F012   ;if not 0 goto F012
 EFFC    LDY     #&EC    ;get first keypress into Y
 EFFE    JSR     &F0CC   ;scan keyboard from &10 (osbyte 122)

 F001    BMI     &F00C   ;if exit is negative goto F00C
 F003    LDA     &EC     ;else make last key the
 F005    STA     &ED     ;first key pressed i.e. rollover

 F007    STX     &EC     ;save X into &EC
 F009    JSR     &F01F   ;set keyboard repeat delay
 F00C    JMP     &EEDA   ;go back to EEDA



*************************************************************************
 *   Key pressed interrupt entry point
*

*************************************************************************
                         ;enters with X=key
 F00F    JSR     &F02A   ;check if key pressed

 F012    LDA     &EC     ;get previous key press
 F014    BNE     &F00C   ;if none back to housekeeping routine
 F016    LDY     #&ED    ;get last keypress into Y
 F018    JSR     &F0CC   ;and scan keyboard
 F01B    BMI     &F00C   ;if negative on exit back to housekeeping
 F01D    BPL     &F007   ;else back to store X and reset keyboard delay
etc.


 *************** Set Autorepeat countdown timer
*************************

 F01F    LDX     #&01    ;set timer to 1
 F021    STX     &E7     ;
 F023    LDX     &0254   ;get next timer value
```

```
F026    STX     &02CA   ;and store it
F029    RTS             ;
```

```
*************** Interrogate Keyboard routine ***********************
                        ;
F02A    LDY     #&03    ;stop Auto scan
F02C    STY     &FE40   ;by writing to system VIA
F02F    LDY     #&7F    ;set bits 0 to 6 of port A to input on bit 7
                        ;output on bits 0 to 6
F031    STY     &FE43   ;
F034    STX     &FE4F   ;write X to Port A system VIA
F037    LDX     &FE4F   ;read back &80 if key pressed (M set)
F03A    RTS             ;and return
```

```
*************************************************************************
 *
*
 *       KEY TRANSLATION TABLES
*
 *
*
 *       7 BLOCKS interspersed with unrelated code
*
*************************************************************************

 *key data block 1

F03B    DB      71,33,34,35,84,38,87,2D,5E,8C
        ;       q ,3 ,4 ,5 ,f4,8 ,f7,- ,^ ,rt
```

```
*************************************************************************
 *
*
 *       OSBYTE 120  Write KEY pressed Data
*
 *
*
 *
*
*************************************************************************
```

```
F045    STY     &EC     ;store Y as latest key pressed
F047    STX     &ED     ;store X as previous key pressed
F049    RTS             ;and exit

 *key data block 2
```

```
F04A    DB      80,77,65,74,37,69,39,30,5F,8E
        ;       f0,w ,e ,t ,7 ,i ,9 ,0 ,_ ,lft

F055    JMP     (&FDFE) ;Jim paged entry vector
F058    JMP     (&FA)   ;

*key data block 3

F05A    DB      31,32,64,72,36,75,6F,70,5B,8F
        ;       1 ,2 ,d ,r ,6 ,u ,o ,p ,[ ,dn



*********************************************************************
 *
*
 * Main entry to keyboard routines
*
 *
*

*********************************************************************

F065    BIT     &D9B7   ;set V and M
F068    JMP     (&0228) ;i.e. KEYV

*key data block 4

F06B    DB      01,61,78,66,79,6A,6B,40,3A,0D
        ;       CL,a ,x ,f ,y ,j ,k ,@ ,: ,RETN  N.B CL=CAPS LOCK

*speech routine data
F075    DB      00,FF,01,02,09,0A

*key data block 5

F07B    DB      02,73,63,67,68,6E,6C,3B,5D,7F
        ;       SL,s ,c ,g ,h ,n ,l ,; ,] ,DEL N.B. SL=SHIFT LOCK



*********************************************************************
 *
*
 *       OSBYTE 131  READ OSHWM  (PAGE in BASIC)
*
 *
*
 *
*

*********************************************************************


F085    LDY     (&0244) ;read current OSHWM
F088    LDX     #&00    ;
F08A    RTS             ;
```

```
 *key data block 6

 F08B    DB              00 ,7A,20   ,76,62,6D,2C,2E,2F,8B
         ;               TAB,Z ,SPACE,V ,b ,m ,, ,. ,/ ,copy

 ***** set input buffer number and flush it ****************************

 F095    LDX     &0241   ;get current input buffer
 F098    JMP     &E1AD   ;flush it

 *key data block 7

 F09B    DB      1B,81,82,83,85,86,88,89,5C,8D
         ;      ESC,f1,f2,f3,f5,f6,f8,f9,\ ,


 F0A5    JMP     (&0220) ;goto eventV handling routine




*************************************************************************
 *
*
 *       OSBYTE 15  FLUSH SELECTED BUFFER CLASS
*
 *
*
 *
*

*************************************************************************

                         ;flush selected buffer
                         ;X=0 flush all buffers
                         ;X>1 flush input buffer

 F0A8    BNE     &F095   ;if X<>1 flush input buffer only
 F0AA    LDX     #&08    ;else load highest buffer number (8)
 F0AC    CLI             ;allow interrupts
 F0AD    SEI             ;briefly!
 F0AE    JSR     &F0B4   ;flush buffer
 F0B1    DEX             ;decrement X to point at next buffer
 F0B2    BPL     &F0AC   ;if X>=0 flush next buffer
                         ;at this point X=255



*************************************************************************
 *
*
 *       OSBYTE 21  FLUSH SPECIFIC BUFFER
*
 *
*
 *
*

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

```
 ;on entry X=buffer number

 F0B4    CPX     #&09    ;is X<9?
 F0B6    BCC     &F098   ;if so flush buffer or else
 F0B8    RTS             ;exit
                         ;

*************************************************************************
 *
*
 *               Issue *HELP to ROMS
*

*************************************************************************
 F0B9    LDX     #&09    ;
 F0BB    JSR     &F168   ;
 F0BE    JSR     &FA4A   ;print following message routine return after
BRK
 F0C1    DB      &0D     ;carriage return
 F0C2    DS      'OS 1.20' ;help message
 F0C9    DB      &0D     ;carriage return
 F0CA    BRK             ;
 F0CB    RTS             ;



*************************************************************************
 *
*
 *       OSBYTE 122  KEYBOARD SCAN FROM &10 (16)
*
 *
*
 *
*

*************************************************************************

                         ;
 F0CC    CLC             ;clear carry
 F0CD    LDX     #&10    ;set X to 10
                         ;



*************************************************************************
 *
*
 *       OSBYTE 121  KEYBOARD SCAN FROM VALUE IN X
*
 *
*
 *
*

*************************************************************************

 F0CF    BCS     &F068   ;if carry set (by osbyte 121) F068
                         ;Jmps via KEYV and hence back to;
```

```
*************************************************************************
 *        Scan Keyboard C=1, V=0 entry via KEYV
*
*************************************************************************

 F0D1    TXA             ;if X is +ve goto F0D9
 F0D2    BPL     &F0D9   ;
 F0D4    JSR     &F02A   ;else interrogate keyboard
 F0D7    BCS     &F12E   ;if carry set F12E to set Auto scan else
 F0D9    PHP             ;push flags
 F0DA    BCC     &F0DE   ;if carry clear goto FODE else
 F0DC    LDY     #&EE    ;set Y so next operation saves to 2cd
 F0DE    STA     &01DF,Y ;can be 2cb,2cc or 2cd
 F0E1    LDX     #&09    ;set X to 9
 F0E3    JSR     &F129   ;select auto scan
 F0E6    LDA     #&7F    ;set port A for input on bit 7 others outputs
 F0E8    STA     &FE43   ;
 F0EB    LDA     #&03    ;stop auto scan
 F0ED    STA     &FE40   ;
 F0F0    LDA     #&0F    ;select non-existent keyboard column F (0-9
only!)
 F0F2    STA     &FE4F   ;
 F0F5    LDA     #&01    ;cancel keyboard interrupt
 F0F7    STA     &FE4D   ;
 F0FA    STX     &FE4F   ;select column X (9 max)
 F0FD    BIT     &FE4D   ;if bit 1 =0 there is no keyboard interrupt so
 F100    BEQ     &F123   ;goto F123
 F102    TXA             ;else put column address in A

 F103    CMP     &01DF,Y ;compare with 1DF+Y
 F106    BCC     &F11E   ;if less then F11E
 F108    STA     &FE4F   ;else select column again
 F10B    BIT     &FE4F   ;and if bit 7 is 0
 F10E    BPL     &F11E   ;then F11E
 F110    PLP             ;else push and pull flags
 F111    PHP             ;
 F112    BCS     &F127   ;and if carry set goto F127
 F114    PHA             ;else Push A
 F115    EOR     &0000,Y ;EOR with EC,ED, or EE depending on Y value
 F118    ASL             ;shift left
 F119    CMP     #&01    ;set carry if = or greater than number holds
EC,ED,EE
 F11B    PLA             ;get back A
 F11C    BCS     &F127   ;if carry set F127
 F11E    CLC             ;else clear carry
 F11F    ADC     #&10    ;add 16
 F121    BPL     &F103   ;and do it again if 0=<result<128
 F123    DEX             ;decrement X
 F124    BPL     &F0E3   ;scan again if greater than 0
 F126    TXA             ;
 F127    TAX             ;
 F128    PLP             ;pull flags

 F129    JSR     &F12E   ;call autoscan
 F12C    CLI             ;allow interrupts
 F12D    SEI             ;disable interrupts
```

```
*************Enable counter scan of keyboard columns ******************
                        ;called from &EEFD, &F129

F12E    LDA     #&0B    ;select auto scan of keyboard
F130    STA     &FE40   ;tell VIA
F133    TXA             ;Get A into X
F134    RTS             ;and return
```

OS SERIES 9
GEOFF COX

```
*************************************************************************
 *
*
 *       OSBYTE 140  *TAPE
*
 *       selects TAPE filing system
*
 *
*
 *
*
 *       OSBYTE 141  *ROM
*
 *       selects ROM filing system
*
 *
*

*************************************************************************


 F135    EOR     #&8C    ;if its *TAPE A=0 *ROM A=1
 F137    ASL             ;double it
 F138    STA     &0247   ;store it in filing system flag store
 F13B    CPX     #&03    ;if X>=3 C set X=3 Z set
 F13D    JMP     &F14B   ;

 ******** set cassette options
****************************************
         ;called after BREAK etc
         ;lower nybble sets sequential access
         ;upper sets load and save options

         ;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


 F140    PHP             ;save flags
 F141    LDA     #&A1    ;set sequential access abort if error, no
messages
 F143    STA     &E3     ;set load/save retry if error, short messages
 F145    LDA     #&19    ;set interblock gap
 F147    STA     &03D1   ;and store it
 F14A    PLP             ;get back flags

 F14B    PHP             ;push flags
 F14C    LDA     #&06    ;get close files command to FSCV
 F14E    JSR     &E031   ;and gosub OSFSC
 F151    LDX     #&06    ;
```

```
 F153    PLP             ;get back flags
 F154    BEQ     &F157   ;if Z set earlier
 F156    DEX             ;do not decrement X
 F157    STX     &C6     ;set current baud rate X=5 300 baud X=6 1200
baud

 ********* reset FILEV,ARGSV,BGETV,BPUTV,GBPBV,FINDV,FSCV
******************
 **********   to F27D, F18E, F4C9, F529, FFA6, F3CA, F1B1
******************

 F159    LDX     #&0E    ;RESET VECTORS FOR FILE RELATED OPERATIONS
 F15B    LDA     &D951,X ;
 F15E    STA     &0211,X ;
 F161    DEX             ;
 F162    BNE     &F15B   ;

 F164    STX     &C2     ;&C2=0 PROGRESS FLAG
 F166    LDX     #&0F    ;set X to make Rom service call &F claim
vectors!




*************************************************************************
 *
*
 *       OSBYTE 143
*
 *       Pass service commands to sideways Roms
*
 *
*

*************************************************************************
                 ;on entry X=command number

 F168    LDA     &F4     ;get current Rom number
 F16A    PHA             ;store it
 F16B    TXA             ;command in A
 F16C    LDX     #&0F    ;set X=15

                         ;send commands loop
 F16E    INC     &02A1,X ;read bit 7 on rom map (no rom has type 254 &FE)
 F171    DEC     &02A1,X ;
 F174    BPL     &F183   ;if not set (+ve result)
 F176    STX     &F4     ;else store rom number in &F4
 F178    STX     &FE30   ;switch in paged ROM
 F17B    JSR     &8003   ;and jump tp service entry
 F17E    TAX             ;on exit put A in X
 F17F    BEQ     &F186   ;if 0 (command recognised by ROM) reset roms &
exit
 F181    LDX     &F4     ;else point to next lower rom
 F183    DEX             ;
 F184    BPL     &F16E   ;and go round loop again

 F186    PLA             ;get back original Rom number
 F187    STA     &F4     ;store it in Ram copy
 F189    STA     &FE30   ;select original page
 F18C    TXA             ;put X back in A
```

```
 F18D    RTS             ;and return


*********************************************************************
 *
*
 *       OSARGS (default) entry point
*
 *       on entry A determines action
*
 *       A=0     save block of memory as a file
*
 *       A=1     write catalogue info for existing file
*
 *       A=2     write load address only for existing file
*
 *       A=3     write execution address only for existing file
*
 *       A=4     write attributes only for existing file
*
 *       A=5     Read catalogue info, return file type in A
*
 *       A=6     Delete named file
*
 *       A=&FF   load the named file if lo byte of Exec address=0 use
*
 *               address in parameter block else files own load address
*
 *       X,Y     point to parameter block
*
 *       bytes   0,1 filename address, 2-5 load,6-9 exec,A-D length or
*
 *               start of data for save, 0E End address /attributes
*
*********************************************************************

 F18E    ORA     #&00    ;is A=00
 F190    BNE     &F1A2   ;if not return
 F192    CPY     #&00    ;is Y=0
 F194    BNE     &F1A2   ;if not return
 F196    LDA     &C6     ;else get current baud rate and zero bit 2
 F198    AND     #&FB    ;C6=5 becomes 1, 6 becomes 2
 F19A    ORA     &0247   ;if cassette selected A=0 else A=2
 F19D    ASL             ;multiply by 2
 F19E    ORA     &0247   ;Or it again
 F1A1    LSR             ;divide by 2
 F1A2    RTS             ;return cassette =0


*********************************************************************
 *
*
 *       FSC     VECTOR  TABLE
*
 *
*
```

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

 F1A3    DW      4C,F5   ; *OPT          (F54C)
 F1A5    DW      1D,F6   ; check EOF     (F61D)
 F1A7    DW      04,F3   ; */            (F304)
 F1A9    DW      0F,E3   ; Bad Command   (E30F) if roms and FS don't want
it
 F1AB    DW      04,F3   ; *RUN          (F304)
 F1AD    DW      2A,F3   ; *CAT          (F32A)
 F1AF    DW      74,E2   ; osbyte 77     (E274)



*********************************************************************
 *       Filing System control entry     OSFSC
*
 *       Entry via 021E FSCV
*
 *       A= index 0 to 7
*
*********************************************************************
         ;on entry A is reason code
         ;A=0    A *OPT command has been used X & Y are the 2 parameters
         ;A=1    EOF is being checked, on entry  X=File handle
                                        on Exit X=FF = EOF exists else 00
         ;A=2    A */ command has been used *RUN the file
         ;A=3    An unrecognised OS command has ben used X,Y point at
command
         ;A=4    A *RUN command has been used X,Y point at filename
         ;A=5    A *CAT cammand has been issued X,Y point to rest of
command
         ;A=6    New filing system about to take over close *SPOOL &*EXEC
files
         ;A=7    Return in X and Y lowest and highest file handle used
         ;A=8    OS about to process *command

 ;A=7 and A=8 are handled by current filing system by changing FSCV

 F1B1    CMP     #&07    ;if A>6
 F1B3    BCS     &F1A2   ;goto F1A2 (RTS)
 F1B5    STX     &BC     ;else save X
 F1B7    ASL             ;A=A*2
 F1B8    TAX             ;X=A to get offset
 F1B9    LDA     &F1A4,X ;get hi byte of address
 F1BC    PHA             ;push it
 F1BD    LDA     &F1A3,X ;get lo byte of address
 F1C0    PHA             ;push it
 F1C1    LDX     &BC     ;get back X
 F1C3    RTS             ;this now jumps to the address got from the
table +1
                         ;the next RTS takes us back to CLI



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

```
 *       LOAD FILE
*
 *
*

********************************************************************************

 F1C4    PHP             ;save flags on stack
 F1C5    PHA             ;save A on stack
 F1C6    JSR     &FB27   ;Set cassette optionsinto (BB),set C7=6
                         ;claim serial system for cassette
 F1C9    LDA     &03C2   ;execution address LO
 F1CC    PHA             ;save A on stack
 F1CD    JSR     &F631   ;search for file
 F1D0    PLA             ;get back A
 F1D1    BEQ     &F1ED   ;if A=0 F1ED
 F1D3    LDX     #&03    ;else X=3
 F1D5    LDA     #&FF    ;A=&FF
 F1D7    PHA             ;save A on stack
 F1D8    LDA     &03BE,X ;get load address
 F1DB    STA     &B0,X   ;store it as current load address
 F1DD    PLA             ;get back A
 F1DE    AND     &B0,X   ;
 F1E0    DEX             ;X=X-1
 F1E1    BPL     &F1D7   ;until all 4 bytes copied

 F1E3    CMP     #&FF    ;if all bytes contain don't contain &FF
 F1E5    BNE     &F1ED   ;continue
 F1E7    JSR     &FAE8   ;else sound bell, reset ACIA & motor off
 F1EA    JMP     &E267   ;'Bad Address' error

 F1ED    LDA     &03CA   ;block flag
 F1F0    LSR             ;set carry from bit 0
 F1F1    PLA             ;get back A
 F1F2    BEQ     &F202   ;if A=0 F202
 F1F4    BCC     &F209   ;if carry clear F209


 ************** LOCKED FILE ROUTINE
************************************

 F1F6    JSR     &FAF2   ;enable second processor and reset serial system

 F1F9    BRK             ;
 F1FA    DB      &E5     ;error number
 F1FC    'Locked'        ;
 F201    BRK             ;


 F202    BCC     &F209   ;if carry clear F209
 F204    LDA     #&03    ;else A=3
 F206    STA     &0258   ;store to cause ESCAPE disable and memory
                         ;clear on break

 F209    LDA     #&30    ;
 F20B    AND     &BB     ;current OPTions
 F20D    BEQ     &F213   ;if options and #&30 =0 ignore error condityion
is set
 F20F    LDA     &C1     ;else get checksum result
 F211    BNE     &F21D   ;and if not 0 F21D
```

```
 F213    TYA             ;A=Y
 F214    PHA             ;save A on stack
 F215    JSR     &FBBB   ;read from second processor if present
 F218    PLA             ;get back A
 F219    TAY             ;Y=A
 F21A    JSR     &F7D5   ;reset flags and check block length
 F21D    JSR     &F9B4   ;load file from tape
 F220    BNE     &F255   ;if not found return to search
 F222    JSR     &FB69   ;increment current block number
 F225    BIT     &03CA   ;block flag
 F228    BMI     &F232   ;if bit 7=1 then this is last block so F232
 F22A    JSR     &F96A   ;else increment current load address
 F22D    JSR     &F77B   ;read block header
 F230    BNE     &F209   ;and goto F209

 ******** store data in OSFILE parameter block
**************************

 F232    LDY     #&0A    ;Y=&0A
 F234    LDA     &CC     ;file length counter lo
 F236    STA     (&C8),Y ;OSFILE parameter  block
 F238    INY             ;Y=Y+1
 F239    LDA     &CD     ;file length counter hi
 F23B    STA     (&C8),Y ;OSFILE parameter  block
 F23D    LDA     #&00    ;A=0
 F23F    INY             ;Y=Y+1
 F240    STA     (&C8),Y ;OSFILE parameter  block
 F242    INY             ;Y=Y+1
 F243    STA     (&C8),Y ;OSFILE parameter  block
 F245    PLP             ;get back flags
 F246    JSR     &FAE8   ;bell, reset ACIA & motor
 F249    BIT     &BA     ;current block flag
 F24B    BMI     &F254   ;return
 F24D    PHP             ;save flags on stack
 F24E    JSR     &FA46   ; print message following call (in this case
NEWLINE!)
 F251    DB      &0D,&00 ;message
 F254    RTS             ;return
         ;
 ************RETRY AFTER A FAILURE ROUTINE
******************************

 F255    JSR     &F637   ;search for a specified block
 F258    BNE     &F209   ;goto F209



 *********** Read Filename using Command Line Interpreter
****************

 ;filename pointed to by X and Y

 F25A    STX     &F2     ;OS filename/command line pointer lo
 F25C    STY     &F3     ;OS filename/command line pointer
 F25E    LDY     #&00    ;Y=0
 F260    JSR     &EA1D   ;initialise string
 F263    LDX     #&00    ;X=0
 F265    JSR     &EA2F   ;GSREAD call
 F268    BCS     &F277   ;if end of character string F277
```

```
 F26A    BEQ     &F274   ;if 0 found F274
 F26C    STA     &03D2,X ;else store character in CFS filename area
 F26F    INX             ;X=X+1
 F270    CPX     #&0B    ;if X<>11
 F272    BNE     &F265   ;then read next
 F274    JMP     &EA8F   ;else Bad String error


 ************* terminate Filename
*************************************

 F277    LDA     #&00    ;terminate filename with 0
 F279    STA     &03D2,X ;
 F27C    RTS             ;return




*******************************************************************************
 *
*
 *       OSFILE ENTRY
*
 *
*

*******************************************************************************

 ;parameter block located by XY
 ;0/1    Address of Filename terminated by &0D
 ;2/4    Load Address of File
 ;6/9    Execution Address of File
 ;A/D    Start address of data for write operations or length of file
 ;       for read operations
 ;E/11   End address of Data; i.e. byte AFTER last byte to be written
 ;       or file attributes
 ;
 ;On Entry action is determined by value in A
 ;
 ;A=0    Save section of memory as named file, write catalogue
information
 ;A=1    Write catalogue information for named file
 ;A=2    Write the LOAD       address (only) for the named File
 ;A=3    Write the EXECUTION  address (only) for the named File
 ;A=4    Write the ATTRIBUTES for the named File
 ;A=5    Read the named files catalogue information Place file type in A
 ;A=6    Delete the named file
 ;A=&FF  Load the named file and read its catalogue information

 F27D    PHA             ;save A on stack
 F27E    STX     &C8     ;osfile block pointer lo
 F280    STY     &C9     ;osfile block pointer hi
 F282    LDY     #&00    ;Y=0
 F284    LDA     (&C8),Y ;OSFILE parameter  block
 F286    TAX             ;X=A
 F287    INY             ;Y=Y+1
 F288    LDA     (&C8),Y ;OSFILE parameter  block
 F28A    TAY             ;Y=A
```

```
 F28B    JSR     &F25A   ;get filename from BUFFER
 F28E    LDY     #&02    ;Y=2

 F290    LDA     (&C8),Y ;copy parameters to Cassette block at 3BE/C5
 F292    STA     &03BC,Y ;from LOAD and EXEC address
 F295    STA     &00AE,Y ;make second copy at B0-B8
 F298    INY             ;Y=Y+1
 F299    CPY     #&0A    ;until Y=10
 F29B    BNE     &F290   ;

 F29D    PLA             ;get back A
 F29E    BEQ     &F2A7   ;if A=0 F2A7
 F2A0    CMP     #&FF    ;else if A<>&FF
 F2A2    BNE     &F254   ;RETURN as cassette has no other options
 F2A4    JMP     &F1C4   ;load file


 ************** Save a file
*****************************************

 F2A7    STA     &03C6   ;zero block number
 F2AA    STA     &03C7   ;zero block number hi

 F2AD    LDA     (&C8),Y ;OSFILE parameter  block
 F2AF    STA     &00A6,Y ;store to Zero page copy (&B0 to &B7)
 F2B2    INY             ;data start and data end address
 F2B3    CPY     #&12    ;until Y=18
 F2B5    BNE     &F2AD   ;
 F2B7    TXA             ;A=X
 F2B8    BEQ     &F274   ;if X=0 no filename found so B274 else BAD
STRING error

 F2BA    JSR     &FB27   ;Set cassette option sinto (BB),set C7=6
                         ;claim serial system for cassette
 F2BD    JSR     &F934   ;prompt to start recording

 F2C0    LDA     #&00    ;A=0
 F2C2    JSR     &FBBD   ;enable 2nd proc. if present and set up osfile
block
 F2C5    JSR     &FBE2   ;set up CFS for write operation
 F2C8    SEC             ;set carry flag
 F2C9    LDX     #&FD    ;X=&FD

 F2CB    LDA     &FFB7,X ;set 03C8/A block length and block flag
 F2CE    SBC     &FFB3,X ;to B4/6-B0/2 the number of pages (blocks) to be
                         ;saved
 F2D1    STA     &02CB,X ;
 F2D4    INX             ;X=X+1
 F2D5    BNE     &F2CB   ;

 F2D7    TAY             ;Y=A
 F2D8    BNE     &F2E8   ;if last byte is non zero F2E8 else
 F2DA    CPX     &03C8   ;compare X with block length
 F2DD    LDA     #&01    ;A=1
 F2DF    SBC     &03C9   ;subtract block length hi
 F2E2    BCC     &F2E8   ;if carry clear F2E8

 F2E4    LDX     #&80    ;X=&80
 F2E6    BNE     &F2F0   ;jump F2F0
```

```
 F2E8    LDA     #&01    ;A=1
 F2EA    STA     &03C9   ;block length hi
 F2ED    STX     &03C8   ;block length
 F2F0    STX     &03CA   ;block flag
 F2F3    JSR     &F7EC   ;write block to Tape
 F2F6    BMI     &F341   ;return if negative
 F2F8    JSR     &F96A   ;increment current load address
 F2FB    INC     &03C6   ;block number
 F2FE    BNE     &F2C8   ;if not 0 loop back again else
 F300    INC     &03C7   ;block number hi
 F303    BNE     &F2C8   ;and loop back again



*********************************************************************
 *
*
 *       *RUN    ENTRY
*
 *
*

*********************************************************************



 F305    JSR     &F25A   ;get filename from BUFFER
 F308    LDX     #&FF    ;X=&FF
 F30A    STX     &03C2   ;execution address
 F30D    JSR     &F1C4   ;load file
 F310    BIT     &027A   ;&FF if tube present
 F313    BPL     &F31F   ;so if not present F31F else
 F315    LDA     &03C4   ;execution address extend
 F318    AND     &03C5   ;execution address extend
 F31B    CMP     #&FF    ;if they are NOT both &FF i.e.for base processor
 F31D    BNE     &F322   ;F322 else
 F31F    JMP     (&03C2) ; RUN file

 F322    LDX     #&C2    ;point to execution address
 F324    LDY     #&03    ;(&3C2)
 F326    LDA     #&04    ;Tube call 4
 F328    JMP     &FBC7   ;and issue to Tube to run file




*********************************************************************
 *
*
 *       *CAT    ENTRY
*
 *
*

*********************************************************************

         ;CASSETTE OPTIONS in &E2
```

```
        ;bit 0  input file open
        ;bit 1  output file open
        ;bit 2,4,5  not used
        ;bit 3  current CATalogue status
        ;bit 6  EOF reached
        ;bit 7  EOF warning given

F32B    LDA     #&08    ;A=8
F32D    JSR     &F344   ;set status bits from A
F330    JSR     &FB27   ;Set cassette options into (BB),set C7=6
                        ;claim serial system for cassette
F333    LDA     #&00    ;A=0
F335    JSR     &F348   ;read data from CFS/RFS
F338    JSR     &FAFC   ;perform read
F33B    LDA     #&F7    ;A=&F7
F33D    AND     &E2     ;clear bit 3 of CFS status bit
F33F    STA     &E2     ;
F341    RTS             ;return
        ;

F342    LDA     #&40    ;set bit 6 of E2 cassette options
F344    ORA     &E2     ;
F346    BNE     &F33F   ;and Jump F33F



********** search routine
*********************************************

F348    PHA             ;save A on stack
F349    LDA     &0247   ;filing system flag 0=CFS 2=RFS
F34C    BEQ     &F359   ;if CFS F359 else
F34E    JSR     &EE13   ;set current Filing System ROM/PHROM
F351    JSR     &EE18   ;get byte from data Romcheck type
F354    BCC     &F359   ;if carry clear F359 else
F356    CLV             ;clear overflow flag
F357    BVC     &F39A   ;JUMP F39A


********** cassette routine*********************************************

F359    JSR     &F77B   ;read block header
F35C    LDA     &03C6   ;block number
F35F    STA     &B4     ;current block no. lo
F361    LDA     &03C7   ;block number hi
F364    STA     &B5     ;current block no. hi
F366    LDX     #&FF    ;X=&FF
F368    STX     &03DF   ;copy of last read block flag
F36B    INX             ;X=X+1
F36C    STX     &BA     ;current block flag
F36E    BEQ     &F376   ;if  0 F376

F370    JSR     &FB69   ;inc. current block no.
F373    JSR     &F77B   ;read block header
F376    LDA     &0247   ;get filing system flag 0=CFS 2=RFS
F379    BEQ     &F37D   ;if CFS F37D
F37B    BVC     &F39A   ;if V clear F39A
F37D    PLA             ;get back A
F37E    PHA             ;save A on stack
```

```
 F37F    BEQ     &F3AE   ;if A=0 F3AE
 F381    JSR     &FA72   ;else check filename header block matches
searched Fn
 F384    BNE     &F39C   ;if so F39C
 F386    LDA     #&30    ;else A=30 to clear all but bits 4/5 of current
OPTions
 F388    AND     &BB     ;current OPTions
 F38A    BEQ     &F39A   ;if 0 F39A else

 F38C    LDA     &03C6   ;block number
 F38F    CMP     &B6     ;next block no. lo
 F391    BNE     &F39C   ;
 F393    LDA     &03C7   ;block number hi
 F396    CMP     &B7     ;next block no. hi
 F398    BNE     &F39C   ;
 F39A    PLA             ;get back A
 F39B    RTS             ;return
         ;
 F39C    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F39F    BEQ     &F3AE   ;if tape F3AE
 F3A1    JSR     &EEAD   ;else set ROM displacement address

 F3A4    LDA     #&FF    ;A=&FF
 F3A6    STA     &03C6   ;block number
 F3A9    STA     &03C7   ;block number hi
 F3AC    BNE     &F370   ;jump F370

 F3AE    BVC     &F3B5   ;if carry clear F3B5
 F3B0    LDA     #&FF    ;A=&FF
 F3B2    JSR     &F7D7   ;set flags
 F3B5    LDX     #&00    ;X=0
 F3B7    JSR     &F9D9   ;report 'DATA?'
 F3BA    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F3BD    BEQ     &F3C3   ;
 F3BF    BIT     &BB     ;current OPTions
 F3C1    BVC     &F3A1   ;long messages not required if BIT 6 =0
 F3C3    BIT     &03CA   ;block flag
 F3C6    BMI     &F3A4   ;if -ve F3A4
 F3C8    BPL     &F370   ;else loop back and do it again




*********************************************************************
 *
*
 *       OSFIND  ENTRY
*
 *       file handling
*
 *
*

*********************************************************************
 ;on entry A determines Action Y may contain file handle or
 ;X/Y point to filename terminated by &0D in memory
 ;A=0    closes file in channel Y if Y=0 closes all files
 ;A=&40  open a file for input  (reading) X/Y points to filename
 ;A=&80  open a file for output (writing) X/Y points to filename
```

```
 ;A=&C0  open a file for input and output (random Access)
 ;ON EXIT Y=0 if no file found else Y=channel number in use for file

                          ;save A X and Y
 F3CA    STA     &BC      ;file status or temporary store
 F3CC    TXA              ;A=X
 F3CD    PHA              ;save X on stack
 F3CE    TYA              ;A=Y
 F3CF    PHA              ;save Y on stack
 F3D0    LDA     &BC      ;file status or temporary store
 F3D2    BNE     &F3F2    ;if A is non zero open a file via F3F2


 ************ close a file
*********************************************

 F3D4    TYA              ;A=Y
 F3D5    BNE     &F3E3    ;if A<> 0 close specified file else close them
all
 F3D7    JSR     &E275    ;close all files via OSBYTE 77
 F3DA    JSR     &F478    ;tidy up
 F3DD    LSR     &E2      ;CFS status byte is shifted left and right to
zero
 F3DF    ASL     &E2      ;bit 0
 F3E1    BCC     &F3EF    ;and if carry clear no input file was open so
F3EF

 F3E3    LSR              ;A contains file handle so shift bit 0 into
carry
 F3E4    BCS     &F3DD    ;if carry set close input file
 F3E6    LSR              ;else shift bit 1 into carry
 F3E7    BCS     &F3EC    ;if carry set close output file
 F3E9    JMP     &FBB1    ;else report 'Channel Error' as CFS can only
support
                          ;1 input and 1 output file

 F3EC    JSR     &F478    ;tidy up
 F3EF    JMP     &F471    ;and exit



 ************ OPEN A FILE
*********************************************

 F3F2    JSR     &F25A    ;get filename from BUFFER
 F3F5    BIT     &BC      ;file status or temporary store
 F3F7    BVC     &F436    ;check A at input if bit 6 not set its an output
file

 ********* Input files
+*********************************************

 F3F9    LDA     #&00     ;else its an input file
 F3FB    STA     &039E    ;BGET buffer offset for next byte
 F3FE    STA     &03DD    ;Expected BGET file block number lo
 F401    STA     &03DE    ;expected BGET file block number hi
 F404    LDA     #&3E     ;A=&3E
 F406    JSR     &F33D    ;CFS status =CFS status AND A
 F409    JSR     &FB1A    ;claim serial system and set OPTions
 F40C    PHP              ;save flags on stack
```

```
 F40D    JSR     &F631   ;search for file
 F410    JSR     &F6B4   ;check protection bit of block status and
respond
 F413    PLP             ;get back flags
 F414    LDX     #&FF    ;X=&FF increment to 0 on next instruction

 F416    INX             ;X=X+1
 F417    LDA     &03B2,X ;get file name and
 F41A    STA     &03A7,X ;store as BGET filename
 F41D    BNE     &F416   ;until end of filename

 F41F    LDA     #&01    ;A=1 to show file open
 F421    JSR     &F344   ;set status bits from A
 F424    LDA     &02EA   ;CFS currently resident file block length lo
 F427    ORA     &02EB   ;CFS currently resident file block length hi
 F42A    BNE     &F42F   ;if block length is 0
 F42C    JSR     &F342   ;set CFS status bit 3 (EOF reached)
                         ;else
 F42F    LDA     #&01    ;A=1
 F431    ORA     &0247   ;filing system flag 0=CFS 2=RFS
 F434    BNE     &F46F   ;and exit after restoring registers


 ******************* open an output
file*********************************

 F436    TXA             ;A=X

 F437    BNE     &F43C   ;if X=0 then zero length filename so
 F439    JMP     &EA8F   ;Bad String error

 F43C    LDX     #&FF    ;X=&FF
 F43E    INX             ;X=X+1
                         ;copy sought filename to header block
 F43F    LDA     &03D2,X ;sought filename
 F442    STA     &0380,X ;BPUT file header block
 F445    BNE     &F43E   ;until A=0 end of filename
 F447    LDA     #&FF    ;A=&FF
 F449    LDX     #&08    ;X=8

 F44B    STA     &038B,X ;set 38C/93 to &FF
 F44E    DEX             ;X=X-1
 F44F    BNE     &F44B   ;

 F451    TXA             ;A=X=0
 F452    LDX     #&14    ;X=14
 F454    STA     &0380,X ;BPUT file header block
 F457    INX             ;X=X+1
 F458    CPX     #&1E    ;this zeros 394/D
 F45A    BNE     &F454   ;

 F45C    ROL     &0397   ;
 F45F    JSR     &FB27   ;Set cassette optionsinto (BB),set C7=6
                         ;claim serial system for cassette
 F462    JSR     &F934   ;prompt to start recording
 F465    JSR     &FAF2   ;enable second processor and reset serial system
 F468    LDA     #&02    ;A=2
 F46A    JSR     &F344   ;set status bits from A
 F46D    LDA     #&02    ;A=2
 F46F    STA     &BC     ;file status or temporary store
```

```
 F471    PLA             ;get back A
 F472    TAY             ;Y=A
 F473    PLA             ;get back A
 F474    TAX             ;X=A
 F475    LDA     &BC     ;file status or temporary store
 F477    RTS             ;return
         ;

 F478    LDA     #&02    ;A=2 clearing all but bit 1 of status byte
 F47A    AND     &E2     ;CFS status byte, with output file open
 F47C    BEQ     &F477   ;if file not open then exit
 F47E    LDA     #&00    ;else A=0
 F480    STA     &0397   ;setting block length to current value of BPUT
offset
 F483    LDA     #&80    ;A=&80
 F485    LDX     &039D   ;get BPUT buffer ofset
 F488    STX     &0396   ;setting block length to current value of BPUT
offset
 F48B    STA     &0398   ;mark current block as last
 F48E    JSR     &F496   ;save block to tape
 F491    LDA     #&FD    ;A=&FD
 F493    JMP     &F33D   ;CFS status =CFS status AND A


 *********** SAVE BLOCK TO TAPE
*****************************************

 F496    JSR     &FB1A   ;claim serial system and set OPTions

 F499    LDX     #&11    ;X=11
 F49B    LDA     &038C,X ;copy header block from 38C-39D
 F49E    STA     &03BE,X ;to 3BE/DF
 F4A1    DEX             ;X=X-1
 F4A2    BPL     &F49B   ;
                         ;X=&FF
 F4A4    STX     &B2     ;current load address high word
 F4A6    STX     &B3     ;current load address high word
 F4A8    INX             ;X=X+1, (X=0)
 F4A9    STX     &B0     ;current load address lo byte set to &00
 F4AB    LDA     #&09    ;A=9 to set current load address at &900
 F4AD    STA     &B1     ;current load address
 F4AF    LDX     #&7F    ;X=&7F
 F4B1    JSR     &FB81   ;copy from 301/C+X to 3D2/C sought filename
 F4B4    STA     &03DF   ;copy of last read block flag
 F4B7    JSR     &FB8E   ;switch Motor On
 F4BA    JSR     &FBE2   ;set up CFS for write operation
 F4BD    JSR     &F7EC   ;write block to Tape
 F4C0    INC     &0394   ;block number lo
 F4C3    BNE     &F4C8   ;
 F4C5    INC     &0395   ;block number hi
 F4C8    RTS             ;return
```

```
*************************************************************************
 *
*
 *
*
 *       OSBGET  get a byte from a file
*
 *
*
 *
*

*************************************************************************
         ;on ENTRY       Y contains channel number
         ;on EXIT        X and Y are preserved C=0 indicates valid
character
         ;               A contains character (or error) A=&FE End Of
File

                         ;push X and Y
 F4C9    TXA             ;A=X
 F4CA    PHA             ;save A on stack
 F4CB    TYA             ;A=Y
 F4CC    PHA             ;save A on stack
 F4CD    LDA     #&01    ;A=1
 F4CF    JSR     &FB9C   ;check conditions for OSBGET are OK
 F4D2    LDA     &E2     ;CFS status byte
 F4D4    ASL             ;shift bit 7 into carry (EOF warning given)
 F4D5    BCS     &F523   ;if carry set F523
 F4D7    ASL             ;shift bit 6 into carry
 F4D8    BCC     &F4E3   ;if clear EOF not reached F4E3
 F4DA    LDA     #&80    ;else A=&80 setting bit 7 of status byte EOF
warning
 F4DC    JSR     &F344   ;set status bits from A
 F4DF    LDA     #&FE    ;A=&FE
 F4E1    BCS     &F51B   ;if carry set F51B

 F4E3    LDX     &039E   ;BGET buffer offset for next byte
 F4E6    INX             ;X=X+1
 F4E7    CPX     &02EA   ;CFS currently resident file block length lo
 F4EA    BNE     &F516   ;read a byte
                         ;else
 F4EC    BIT     &02EC   ;block flag of currently resident block
 F4EF    BMI     &F513   ;if bit 7=1 this is the last block so F513 else
 F4F1    LDA     &02ED   ;last character of currently resident block
 F4F4    PHA             ;save A on stack
 F4F5    JSR     &FB1A   ;claim serial system and set OPTions
 F4F8    PHP             ;save flags on stack
 F4F9    JSR     &F6AC   ;read in a new block
 F4FC    PLP             ;get back flags
 F4FD    PLA             ;get back A
 F4FE    STA     &BC     ;file status or temporary store
 F500    CLC             ;clear carry flag
 F501    BIT     &02EC   ;block flag of currently resident block
 F504    BPL     &F51D   ;if not last block (bit 7=0)
 F506    LDA     &02EA   ;CFS currently resident file block length lo
 F509    ORA     &02EB   ;CFS currently resident file block length hi
 F50C    BNE     &F51D   ;if block size not 0 F51D else
 F50E    JSR     &F342   ;set CFS status bit 6 (EOF reached)
```

```
 F511    BNE     &F51D   ;goto F51D

 F513    JSR     &F342   ;set CFS status bit 6 (EOF reached)
 F516    DEX             ;X=X-1
 F517    CLC             ;clear carry flag
 F518    LDA     &0A00,X ;read byte from cassette buffer

 F51B    STA     &BC     ;file status or temporary store
 F51D    INC     &039E   ;BGET buffer offset for next byte
 F520    JMP     &F471   ;exit via F471


 F523    BRK             ;
 F524    DB      &DF     ;error number
 F525    DB      'EOF'   ;
 F528    BRK             ;


*************************************************************************
 *
*
 *
*
 *       OSBPUT  WRITE A BYTE TO FILE
*
 *
*
 *
*

*************************************************************************
 ;ON ENTRY  Y contains channel number A contains byte to be written

 F529    STA     &C4     ;store A in temorary store
 F52B    TXA             ;and stack X and Y
 F52C    PHA             ;save  on stack
 F52D    TYA             ;A=Y
 F52E    PHA             ;save on stack

 F52F    LDA     #&02    ;A=2
 F531    JSR     &FB9C   ;check conditions necessary for OSBPUT are OK
 F534    LDX     &039D   ;BPUT buffer offset for next byte
 F537    LDA     &C4     ;get back original value of A
 F539    STA     &0900,X ;Cassette buffer
 F53C    INX             ;X=X+1
 F53D    BNE     &F545   ;if not 0 F545, otherwise buffer is full so
 F53F    JSR     &F496   ;save block to tape
 F542    JSR     &FAF2   ;enable second processor and reset serial system
 F545    INC     &039D   ;BPUT buffer offset for next byte
 F548    LDA     &C4     ;get back A
 F54A    JMP     &F46F   ;and exit



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

```
 *
*
 *       OSBYTE 139      Select file options
*
 *
*
 *
*

*********************************************************************
 ;ON ENTRY  Y contains option value  X contains option No. see *OPT X,Y
 ;this applies largely to CFS LOAD SAVE CAT and RUN
 ;X=1    is message switch
 ;       Y=0     no messages
 ;       Y=1     short messages
 ;       Y=2     gives detailed information on load and execution
addresses

 ;X=2    is error handling
 ;       Y=0     ignore errors
 ;       Y=1     prompt for a retry
 ;       Y=2     abort operation

 ;X=3    is interblock gap for BPUT# and PRINT#
 ;       Y=0-127 set gap in 1/10ths Second
 ;       Y > 127 use default values


 F54D    TXA             ;A=X
 F54E    BEQ     &F57E   ;if A=0 F57E
 F550    CPX     #&03    ;if X=3
 F552    BEQ     &F573   ;F573 to set interblock gap
 F554    CPY     #&03    ;else if Y>2 then BAD COMMAND error
 F556    BCS     &F55E   ;
 F558    DEX             ;X=X-1
 F559    BEQ     &F561   ;i.e. if X=1 F561 message control
 F55B    DEX             ;X=X-1
 F55C    BEQ     &F568   ;i.e. if X=2 F568 error response
 F55E    JMP     &E310   ;else E310 to issue Bad Command error



 ********** message control
*******************************************

 F561    LDA     #&33    ;to set lower two bits of each nybble as mask
 F563    INY             ;Y=Y+1
 F564    INY             ;Y=Y+1
 F565    INY             ;Y=Y+1
 F566    BNE     &F56A   ;goto F56A



 ********** error response ***********************************************

 F568    LDA     #&CC    ;setting top two bits of each nybble as mask
 F56A    INY             ;Y=Y+1
 F56B    AND     &E3     ;clear lower two bits of each nybble
 F56D    ORA     &F581,Y ;or with table value
```

```
F570    STA     &E3     ;store it in &E3
F572    RTS             ;return

        ;setting of &E3
        ;
        ;lower nybble sets LOAD options
        ;upper sets SAVE options

        ;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


***********set interblock gap
****************************************

F573    TYA             ;A=Y
F574    BMI     &F578   ;if Y>127 use default values
F576    BNE     &F57A   ;if Y<>0 skip next instruction
F578    LDA     #&19    ;else A=&19

F57A    STA     &03D1   ;sequential block gap
F57D    RTS             ;return
        ;
F57E    TAY             ;Y=A
F57F    BEQ     &F56D   ;jump to F56D


********** DEFAULT OPT VALUES TABLE
**********************************

F581    DB      &A1     ;%1010 0001
F582    DB      &00     ;%0000 0000
F583    DB      &22     ;%0010 0010
F584    DB      &11     ;%0001 0001
F585    DB      &00     ;%0000 0000
F586    DB      &88     ;%1000 1000
F587    DB      &CC     ;%1100 1100

F588    DEC     &C0     ;filing system buffer flag
F58A    LDA     &0247   ;filing system flag 0=CFS 2=RFS
F58D    BEQ     &F596   ;if CFS F596

F58F    JSR     &EE51   ;read RFS data rom or Phrom
F592    TAY             ;Y=A
F593    CLC             ;clear carry flag
F594    BCC     &F5B0   ;jump to F5B0

F596    LDA     &FE08   ;ACIA status register
F599    PHA             ;save A on stack
F59A    AND     #&02    ;clear all but bits 0,1 A=(0-3)
```

```
 F59C    BEQ     &F5A9   ;if 0 F5A9 transmit data register full or RDR
empty
 F59E    LDY     &CA     ;
 F5A0    BEQ     &F5A9   ;
 F5A2    PLA             ;get back A
 F5A3    LDA     &BD     ;character temporary storage
 F5A5    STA     &FE09   ;ACIA transmit data register
 F5A8    RTS             ;return
         ;
 F5A9    LDY     &FE09   ;read ACIA recieve data register
 F5AC    PLA             ;get back A
 F5AD    LSR             ;bit 2 to carry (data carrier detect)
 F5AE    LSR             ;
 F5AF    LSR             ;

 F5B0    LDX     &C2     ;progress flag
 F5B2    BEQ     &F61D   ;if &C2=0 exit
 F5B4    DEX             ;X=X-1
 F5B5    BNE     &F5BD   ;if &C2>1 then F5BD
 F5B7    BCC     &F61D   ;else if carrier tone from cassette detected
exit

 F5B9    LDY     #&02    ;Y=2

 F5BB    BNE     &F61B   ;
 F5BD    DEX             ;X=X-1
 F5BE    BNE     &F5D3   ;if &C2>2
 F5C0    BCS     &F61D   ;if carrier tone from cassette not detected
exit
 F5C2    TYA             ;A=Y
 F5C3    JSR     &FB78   ;set (BE/C0) to 0
 F5C6    LDY     #&03    ;Y=3
 F5C8    CMP     #&2A    ;is A= to synchronising byte &2A?
 F5CA    BEQ     &F61B   ;if so F61B
 F5CC    JSR     &FB50   ;control cassette system
 F5CF    LDY     #&01    ;Y=1
 F5D1    BNE     &F61B   ;goto F61B
 F5D3    DEX             ;X=X-1
 F5D4    BNE     &F5E2   ;if &C2>3
 F5D6    BCS     &F5DC   ;
 F5D8    STY     &BD     ;get character read into Y
 F5DA    BEQ     &F61D   ;if 0 exit via F61D
 F5DC    LDA     #&80    ;else A=&80
 F5DE    STA     &C0     ;filing system buffer flag
 F5E0    BNE     &F61D   ;and exit

 F5E2    DEX             ;X=X-1
 F5E3    BNE     &F60E   ;if &C2>4 F60E
 F5E5    BCS     &F616   ;if carry set F616
 F5E7    TYA             ;A=Y
 F5E8    JSR     &F7B0   ;perform CRC
 F5EB    LDY     &BC     ;file status or temporary store
 F5ED    INC     &BC     ;file status or temporary store
 F5EF    BIT     &BD     ;if bit 7 set this is the last byte read
 F5F1    BMI     &F600   ;so F600
 F5F3    JSR     &FBD3   ;check if second processor file test tube
prescence
 F5F6    BEQ     &F5FD   ;if return with A=0 F5FD
 F5F8    STX     &FEE5   ;Tube FIFO3
 F5FB    BNE     &F600   ;
```

```
F5FD    TXA             ;A=X restore value
F5FE    STA     (&B0),Y ;store to current load address
F600    INY             ;Y=Y+1
F601    CPY     &03C8   ;block length
F604    BNE     &F61D   ;exit
F606    LDA     #&01    ;A=1
F608    STA     &BC     ;file status or temporary store
F60A    LDY     #&05    ;Y=5
F60C    BNE     &F61B   ;exit

F60E    TYA             ;A=Y
F60F    JSR     &F7B0   ;perform CRC
F612    DEC     &BC     ;file status or temporary store
F614    BPL     &F61D   ;exit

F616    JSR     &FB46   ;reset ACIA
F619    LDY     #&00    ;Y=0
F61B    STY     &C2     ;progress flag
F61D    RTS             ;return



*************************************************************************
 *
*
 *       OSBYTE 127 check for end of file
*
 *
*

*************************************************************************
        ;
F61E    PHA             ;save A on stack
F61F    TYA             ;A=Y
F620    PHA             ;save Y on stack
F621    TXA             ;A=X to put X into Y
F622    TAY             ;Y=A
F623    LDA     #&03    ;A=3
F625    JSR     &FB9C   ;confirm file is open
F628    LDA     &E2     ;CFS status byte
F62A    AND     #&40    ;
F62C    TAX             ;X=A
F62D    PLA             ;get back A
F62E    TAY             ;Y=A
F62F    PLA             ;get back A
F630    RTS             ;return
        ;
F631    LDA     #&00    ;A=0
F633    STA     &B4     ;current block no. lo
F635    STA     &B5     ;current block no. hi
F637    LDA     &B4     ;current block no. lo
F639    PHA             ;save A on stack
F63A    STA     &B6     ;next block no. lo
F63C    LDA     &B5     ;current block no. hi
F63E    PHA             ;save A on stack
F63F    STA     &B7     ;next block no. hi
F641    JSR     &FA46   ;print message following call

F644    DB      'Searching';
```

```
F64C    DB      &0D     ;newline
F64E    BRK             ;

F64F    LDA     #&FF    ;A=&FF
F651    JSR     &F348   ;read data from CFS/RFS
F654    PLA             ;get back A
F655    STA     &B5     ;current block no. hi
F657    PLA             ;get back A
F658    STA     &B4     ;current block no. lo
F65A    LDA     &B6     ;next block no. lo
F65C    ORA     &B7     ;next block no. hi
F65E    BNE     &F66D   ;
F660    STA     &B4     ;current block no. lo
F662    STA     &B5     ;current block no. hi
F664    LDA     &C1     ;checksum result
F666    BNE     &F66D   ;
F668    LDX     #&B1    ;current load address
F66A    JSR     &FB81   ;copy from 301/C+X to 3D2/C sought filename
F66D    LDA     &0247   ;filing system flag 0=CFS 2=RFS
F670    BEQ     &F685   ;if cassette F685
F672    BVS     &F685   ;

F674    BRK             ;
F675    DB      &D6             ;Error number
F676    DB      'File Not found'
F684    BRK             ;

F685    LDY     #&FF    ;Y=&FF
F687    STY     &03DF   ;copy of last read block flag
F68A    RTS             ;return
        ;

********        * EXEC  0
*********************************************

F68B    LDA     #&00    ;A=0



*************************************************************************
 *
*
 *       * EXEC
*
 *
*

*************************************************************************

F68D    PHP             ;save flags on stack
F68E    STY     &E6     ;&E6=Y
F690    LDY     &0256   ;EXEC file handle
F693    STA     &0256   ;EXEC file handle
F696    BEQ     &F69B   ;if not 0 close file via OSFIND
F698    JSR     OSFIND  ;
F69B    LDY     &E6     ;else Y= original Y
F69D    PLP             ;get back flags
F69E    BEQ     &F6AB   ;if A=0 on entry exit else
F6A0    LDA     #&40    ;A=&40
```

```
 F6A2    JSR     OSFIND  ;to open an input file
 F6A5    TAY             ;Y=A
 F6A6    BEQ     &F674   ;If Y=0 'File not found' else store
 F6A8    STA     &0256   ;EXEC file handle
 F6AB    RTS             ;return

 ******* read a block
**********************************************

 F6AC    LDX     #&A6    ;X=&A6
 F6AE    JSR     &FB81   ;copy from 301/C+X to 3D2/C sought filename
 F6B1    JSR     &F77B   ;read block header
 F6B4    LDA     &03CA   ;block flag
 F6B7    LSR             ;A=A/2 bit 0 into carry to check for locked file
 F6B8    BCC     &F6BD   ;if not set then skip next instruction
 F6BA    JMP     &F1F6   ;'locked' file routine

 F6BD    LDA     &03DD   ;Expected BGET file block number lo
 F6C0    STA     &B4     ; current block no. lo
 F6C2    LDA     &03DE   ;expected BGET file block number hi
 F6C5    STA     &B5     ;current block no. hi
 F6C7    LDA     #&00    ;A=0
 F6C9    STA     &B0     ;current load address
 F6CB    LDA     #&0A    ;A=&A setting current load address to the
CFS/RFS
 F6CD    STA     &B1     ;current load address buffer at &A00
 F6CF    LDA     #&FF    ;A=&FF to set other 2 bytes
 F6D1    STA     &B2     ;current load address high word
 F6D3    STA     &B3     ;current load address high word
 F6D5    JSR     &F7D5   ;reset flags
 F6D8    JSR     &F9B4   ;load file from tape
 F6DB    BNE     &F702   ;if return non zero F702 else
 F6DD    LDA     &0AFF   ;get last character from input buffer
 F6E0    STA     &02ED   ;last character currently resident block
 F6E3    JSR     &FB69   ;inc. current block no.
 F6E6    STX     &03DD   ;expected BGET file block number lo
 F6E9    STY     &03DE   ;expected BGET file block number hi
 F6EC    LDX     #&02    ;X=2
 F6EE    LDA     &03C8,X ;read bytes from block flag/block length
 F6F1    STA     &02EA,X ;store into current values of above
 F6F4    DEX             ;X=X-1
 F6F5    BPL     &F6EE   ;until X=-1 (&FF)

 F6F7    BIT     &02EC   ;block flag of currently resident block
 F6FA    BPL     &F6FF   ;
 F6FC    JSR     &F249   ;print newline if needed
 F6FF    JMP     &FAF2   ;enable second processor and reset serial system
 F702    JSR     &F637   ;search for a specified block
 F705    BNE     &F6B4   ;if NE check for locked condition else
 F707    CMP     #&2A    ;is it Synchronising byte &2A?
 F709    BEQ     &F742   ;if so F742
 F70B    CMP     #&23    ;else is it &23 (header substitute in ROM files)
 F70D    BNE     &F71E   ;if not BAD ROM error

 F70F    INC     &03C6   ;block number
 F712    BNE     &F717   ;
 F714    INC     &03C7   ;block number hi
 F717    LDX     #&FF    ;X=&FF
 F719    BIT     &D9B7   ;to set V & M
 F71C    BNE     &F773   ;and jump (ALWAYS!!) to F773
```

```
 F71E    LDA     #&F7    ;clear bit 3 of RFS status (current CAT status)
 F720    JSR     &F33D   ;RFS status =RFS status AND A

 F723    BRK             ;and cause error
 F724    DB      &D7     ;error number
 F725    DB      'Bad Rom'
 F72C    BRK             ;


 **********: pick up a header
*****************************************

 F72D    LDY     &FF     ;get ESCAPE flag
 F72F    JSR     &FB90   ;switch Motor on
 F732    LDA     #&01    ;A=1
 F734    STA     &C2     ;progress flag
 F736    JSR     &FB50   ;control serial system
 F739    JSR     &F995   ;confirm ESC not set and CFS not executing
 F73C    LDA     #&03    ;A=3
 F73E    CMP     &C2     ;progress flag
 F740    BNE     &F739   ;back until &C2=3

 F742    LDY     #&00    ;Y=0
 F744    JSR     &FB7C   ;zero checksum bytes
 F747    JSR     &F797   ;get character from file and do CRC
 F74A    BVC     &F766   ;if V clear on exit F766
 F74C    STA     &03B2,Y ;else store
 F74F    BEQ     &F757   ;or if A=0 F757
 F751    INY             ;Y=Y+1
 F752    CPY     #&0B    ;if Y<>&B
 F754    BNE     &F747   ;go back for next character
 F756    DEY             ;Y=Y-1

 F757    LDX     #&0C    ;X=12
 F759    JSR     &F797   ;get character from file and do CRC
 F75C    BVC     &F766   ;if V clear on exit F766
 F75E    STA     &03B2,X ;else store byte
 F761    INX             ;X=X+1
 F762    CPX     #&1F    ;if X<>31
 F764    BNE     &F759   ;goto F759

 F766    TYA             ;A=Y
 F767    TAX             ;X=A
 F768    LDA     #&00    ;A=0
 F76A    STA     &03B2,Y ;store it
 F76D    LDA     &BE     ;CRC workspace
 F76F    ORA     &BF     ;CRC workspace
 F771    STA     &C1     ;Checksum result
 F773    JSR     &FB78   ;set (BE/C0) to 0
 F776    STY     &C2     ;progress flag
 F778    TXA             ;A=X
 F779    BNE     &F7D4   ;
 F77B    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F77E    BEQ     &F72D   ;if cassette F72D
 F780    JSR     &EE51   ;read RFS data rom or Phrom
 F783    CMP     #&2B    ;is it ROM file terminator?
 F785    BNE     &F707   ;if not F707
```

```
 ********* terminator found
*****************************************

 F787    LDA     #&08    ;A=8 isolating bit 3 CAT status
 F789    AND     &E2     ;CFS status byte
 F78B    BEQ     &F790   ;if clera skip next instruction
 F78D    JSR     &F24D   ;print CR if CFS not operational
 F790    JSR     &EE18   ;get byte from data Rom
 F793    BCC     &F780   ;if carry set F780
 F795    CLV             ;clear overflow flag
 F796    RTS             ;return


 ****************  get character from file and do CRC
******************
        ;
 F797    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F79A    BEQ     &F7AD   ;if cassette F7AD
 F79C    TXA             ;A=X to save X and Y
 F79D    PHA             ;save X on stack
 F79E    TYA             ;A=Y
 F79F    PHA             ;save Y on stack
 F7A0    JSR     &EE51   ;read RFS data rom or Phrom
 F7A3    STA     &BD     ;put it in temporary storage
 F7A5    LDA     #&FF    ;A=&FF
 F7A7    STA     &C0     ;filing system buffer flag
 F7A9    PLA             ;get back Y
 F7AA    TAY             ;Y=A
 F7AB    PLA             ;get back X
 F7AC    TAX             ;X=A
 F7AD    JSR     &F884   ;check for Escape and loop till bit 7 of FS
buffer
                         ;flag=1


 ************************** perform CRC
*********************************

 F7B0    PHP             ;save flags on stack
 F7B1    PHA             ;save A on stack
 F7B2    SEC             ;set carry flag
 F7B3    ROR     &CB     ;CRC Bit counter
 F7B5    EOR     &BF     ;CRC workspace
 F7B7    STA     &BF     ;CRC workspace
 F7B9    LDA     &BF     ;CRC workspace
 F7BB    ROL             ;A=A*2 C=bit 7
 F7BC    BCC     &F7CA   ;
 F7BE    ROR             ;A=A/2
 F7BF    EOR     #&08    ;
 F7C1    STA     &BF     ;CRC workspace
 F7C3    LDA     &BE     ;CRC workspace
 F7C5    EOR     #&10    ;
 F7C7    STA     &BE     ;CRC workspace
 F7C9    SEC             ;set carry flag

 F7CA    ROL     &BE     ;CRC workspace
 F7CC    ROL     &BF     ;CRC workspace
 F7CE    LSR     &CB     ;CRC Bit counter
 F7D0    BNE     &F7B9   ;
```

```
 F7D2    PLA             ;get back A
 F7D3    PLP             ;get back flags
 F7D4    RTS             ;return
         ;


 F7D5    LDA     #&00    ;A=0
 F7D7    STA     &BD     ;&BD=character temporary storage buffer=0
 F7D9    LDX     #&00    ;X=0
 F7DB    STX     &BC     ;file status or temporary store
 F7DD    BVC     &F7E9   ;
 F7DF    LDA     &03C8   ;block length
 F7E2    ORA     &03C9   ;block length hi
 F7E5    BEQ     &F7E9   ;if 0 F7E9

 F7E7    LDX     #&04    ;else X=4
 F7E9    STX     &C2     ;filename length/progress flag
 F7EB    RTS             ;return



 *************** SAVE A BLOCK
****************************************

 F7EC    PHP             ;save flags on stack
 F7ED    LDX     #&03    ;X=3
 F7EF    LDA     #&00    ;A=0
 F7F1    STA     &03CB,X ;clear 03CB/E (RFS EOF+1?)
 F7F4    DEX             ;X=X-1
 F7F5    BPL     &F7F1   ;

 F7F7    LDA     &03C6   ;block number
 F7FA    ORA     &03C7   ;block number hi
 F7FD    BNE     &F804   ;if block =0 F804 else
 F7FF    JSR     &F892   ;generate a 5 second delay
 F802    BEQ     &F807   ;goto F807


 F804    JSR     &F896   ;generate delay set by interblock gap
 F807    LDA     #&2A    ;A=&2A
 F809    STA     &BD     ;store it in temporary file
 F80B    JSR     &FB78   ;set (BE/C0) to 0
 F80E    JSR     &FB4A   ;set ACIA control register
 F811    JSR     &F884   ;check for Escape and loop till bit 7 of FS
buffer
                         ;flag=1
 F814    DEY             ;Y=Y-1
 F815    INY             ;Y=Y+1
 F816    LDA     &03D2,Y ;move sought filename
 F819    STA     &03B2,Y ;into filename block
 F81C    JSR     &F875   ;transfer byte to CFS and do CRC
 F81F    BNE     &F815   ;if filename not complet then do it again


 ******: deal with rest of header
****************************************

 F821    LDX     #&0C    ;X=12
 F823    LDA     &03B2,X ;get filename byte
 F826    JSR     &F875   ;transfer byte to CFS and do CRC
```

```
 F829    INX             ;X=X+1
 F82A    CPX     #&1D    ;until X=29
 F82C    BNE     &F823   ;

 F82E    JSR     &F87B   ;save checksum to TAPE reset buffer flag
 F831    LDA     &03C8   ;block length
 F834    ORA     &03C9   ;block length hi
 F837    BEQ     &F855   ;if 0 F855

 F839    LDY     #&00    ;else Y=0
 F83B    JSR     &FB7C   ;zero checksum bytes
 F83E    LDA     (&B0),Y ;get a data byte
 F840    JSR     &FBD3   ;check if second processor file test tube
presence
 F843    BEQ     &F848   ;if not F848 else

 F845    LDX     &FEE5   ;Tube FIFO3

 F848    TXA             ;A=X
 F849    JSR     &F875   ;transfer byte to CFS and do CRC
 F84C    INY             ;Y=Y+1
 F84D    CPY     &03C8   ;block length
 F850    BNE     &F83E   ;
 F852    JSR     &F87B   ;save checksum to TAPE reset buffer flag
 F855    JSR     &F884   ;check for Escape and loop till bit 7 of FS
buffer
                         ;flag=1
 F858    JSR     &F884   ;check for Escape and loop till bit 7 of FS
buffer
                         ;flag=1
 F85B    JSR     &FB46   ;reset ACIA

 F85E    LDA     #&01    ;A=1
 F860    JSR     &F898   ;generate 0.1 * A second delay
 F863    PLP             ;get back flags
 F864    JSR     &F8B9   ;update block flag, PRINT filename (& address if
reqd)
 F867    BIT     &03CA   ;block flag
 F86A    BPL     &F874   ;is this last block (bit 7 set)?
 F86C    PHP             ;save flags on stack
 F86D    JSR     &F892   ;generate a 5 second delay
 F870    JSR     &F246   ;sound bell and abort
 F873    PLP             ;get back flags
 F874    RTS             ;return


 ****************** transfer byte to CFS and do CRC
**********************
         ;
 F875    JSR     &F882   ;save byte to buffer, transfer to CFS & reset
flag
 F878    JMP     &F7B0   ;perform CRC


 ***************** save checksum to TAPE reset buffer flag
***************

 F87B    LDA     &BF     ;CRC workspace
 F87D    JSR     &F882   ;save byte to buffer, transfer to CFS & reset
flag
```

```
 F880    LDA     &BE     ;CRC workspace


 ************** save byte to buffer, transfer to CFS & reset flag
********

 F882    STA     &BD     ;store A in temporary buffer


 ***** check for Escape and loop till bit 7 of FS buffer flag=1
**********

 F884    JSR     &F995   ;confirm ESC not set and CFS not executing
 F887    BIT     &C0     ;filing system buffer flag
 F889    BPL     &F884   ;loop until bit 7 of &C0 is set

 F88B    LDA     #&00    ;A=0
 F88D    STA     &C0     ;filing system buffer flag
 F88F    LDA     &BD     ;get temporary store byte
 F891    RTS             ;return
         ;


 ****************** generate a 5 second delay
**************************

 F892    LDA     #&32    ;A=50
 F894    BNE     &F898   ;generate delay 100ms *A (5 seconds)



 *************** generate delay set by interblock gap
********************

 F896    LDA     &C7     ;get current interblock flag


 *************** generate delay
****************************************

 F898    LDX     #&05    ;X=5
 F89A    STA     &0240   ;CFS timeout counter
 F89D    JSR     &F995   ;confirm ESC not set and CFS not executing
 F8A0    BIT     &0240   ;CFS timeout counter (decremented each 20ms)
 F8A3    BPL     &F89D   ;if +ve F89D
 F8A5    DEX             ;X=X-1
 F8A6    BNE     &F89A   ;
 F8A8    RTS             ;return
         ;


 ************: generate screen reports
**********************************

 F8A9    LDA     &03C6   ;block number
 F8AC    ORA     &03C7   ;block number hi
 F8AF    BEQ     &F8B6   ;if 0 F8B6
 F8B1    BIT     &03DF   ;copy of last read block flag
 F8B4    BPL     &F8B9   ;update block flag, PRINT filename (& address if
reqd)
```

```
 F8B6    JSR     &F249   ;print newline if needed


 ************** update block flag, PRINT filename (& address if reqd)
****

 F8B9    LDY     #&00    ;Y=0
 F8BB    STY     &BA     ;current block flag
 F8BD    LDA     &03CA   ;block flag
 F8C0    STA     &03DF   ;copy of last read block flag
 F8C3    JSR     &E7DC   ;check if free to print message
 F8C6    BEQ     &F933   ;if A=0 on return Cassette system is busy
 F8C8    LDA     #&0D    ;else A=&0D :carriage return
 F8CA    JSR     OSWRCH  ;print it (note no linefeed as its via OSWRCH)
 F8CD    LDA     &03B2,Y ;get byte form filename
 F8D0    BEQ     &F8E2   ;if 0 filename is ended
 F8D2    CMP     #&20    ;if <SPACE
 F8D4    BCC     &F8DA   ;F8DA
 F8D6    CMP     #&7F    ;if less than DELETE
 F8D8    BCC     &F8DC   ;its a printable character for F8DC else


 ********************Control characters in RFS/CFS filename
******************

 F8DA    LDA     #&3F    ;else A='?'
 F8DC    JSR     OSWRCH  ;and print it

 F8DF    INY             ;Y=Y+1
 F8E0    BNE     &F8CD   ;back to get rest of filename


 ***************** end of filename
**************************************

 F8E2    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F8E5    BEQ     &F8EB   ;if cassette F8EB
 F8E7    BIT     &BB     ;test current OPTions
 F8E9    BVC     &F933   ;if bit 6 clear no,long messages needed F933
 F8EB    JSR     &F991   ;print a space
 F8EE    INY             ;Y=Y+1
 F8EF    CPY     #&0B    ;if Y<11 then
 F8F1    BCC     &F8E2   ;loop again to fill out filename with spaces

 F8F3    LDA     &03C6   ;block number
 F8F6    TAX             ;X=A
 F8F7    JSR     &F97A   ;print ASCII equivalent of hex byte
 F8FA    BIT     &03CA   ;block flag
 F8FD    BPL     &F933   ;if not end of file return
 F8FF    TXA             ;A=X
 F900    CLC             ;clear carry flag
 F901    ADC     &03C9   ;block length hi
 F904    STA     &CD     ;file length counter hi
 F906    JSR     &F975   ;print space + ASCII equivalent of hex byte
 F909    LDA     &03C8   ;block length
 F90C    STA     &CC     ;file length counter lo
 F90E    JSR     &F97A   ;print ASCII equivalent of hex byte
 F911    BIT     &BB     ;current OPTions
 F913    BVC     &F933   ;if bit 6 clear no long messages required so
F933
```

```
 F915    LDX     #&04    ;X=4
 F917    JSR     &F991   ;print a space
 F91A    DEX             ;X=X-1
 F91B    BNE     &F917   ;loop to print 4 spaces

 F91D    LDX     #&0F    ;X=&0F to point to load address
 F91F    JSR     &F927   ;print 4 bytes from CFS block header
 F922    JSR     &F991   ;print a space
 F925    LDX     #&13    ;X=&13 point to Execution address


 ************** print 4 bytes from CFS block header
**********************

 F927    LDY     #&04    ;loop pointer
 F929    LDA     &03B2,X ;block header
 F92C    JSR     &F97A   ;print ASCII equivalent of hex byte
 F92F    DEX             ;X=X-1
 F930    DEY             ;Y=Y-1
 F931    BNE     &F929   ;

 F933    RTS             ;return

         ;
 *********** print prompt for SAVE on TAPE
******************************

 F934    LDA     &0247   ;filing system flag 0=CFS 2=RFS
 F937    BEQ     &F93C   ;if cassette F93C
 F939    JMP     &E310   ;else 'Bad Command error message'
 F93C    JSR     &FB8E   ;switch Motor On
 F93F    JSR     &FBE2   ;set up CFS for write operation
 F942    JSR     &E7DC   ;check if free to print message
 F945    BEQ     &F933   ;if not exit else
 F947    JSR     &FA46   ; print message following call

 F94A    DB      'RECORD then RETURN';
 F95C    BRK             ;

 F95D    JSR     &F995   ;confirm CFS not operating, nor ESCAPE flag set



 ************ wait for RETURN key to be pressed
**************************

 F960    JSR     OSRDCH  ;wait for keypress
 F963    CMP     #&0D    ;is it &0D (RETURN)
 F965    BNE     &F95D   ;no then do it again

 F967    JMP     OSNEWL  ;output Carriage RETURN and LINE FEED



 ************* increment current load address
****************************

 F96A    INC     &B1     ;current load address
 F96C    BNE     &F974   ;
```

```
 F96E    INC     &B2     ;current load address high word
 F970    BNE     &F974   ;
 F972    INC     &B3     ;current load address high word
 F974    RTS             ;return
         ;
 ************* print a space + ASCII equivalent of hex byte
**************

 F975    PHA             ;save A on stack
 F976    JSR     &F991   ;print a space
 F979    PLA             ;get back A


 ************** print ASCII equivalent of hex byte
**********************

 F97A    PHA             ;save A on stack
 F97B    LSR             ;/16 to put high nybble in lo
 F97C    LSR             ;
 F97D    LSR             ;
 F97E    LSR             ;
 F97F    JSR     &F983   ;print its ASCII equivalent
 F982    PLA             ;get back A

 F983    CLC             ;clear carry flag
 F984    AND     #&0F    ;clear high nybble
 F986    ADC     #&30    ;Add &30 to convert 0-9 to ASCII A-F to : ; < =
> ?
 F988    CMP     #&3A    ;if A< ASC(':')
 F98A    BCC     &F98E   ;goto F98E
 F98C    ADC     #&06    ;else add 7 to convert : ; < = > ? to A B C D E
F

 F98E    JMP     OSWRCH  ;print character and return



 ******************** print a space
************************************

 F991    LDA     #&20    ;A=' '
 F993    BNE     &F98E   ;goto F98E to print it



 ******************** confirm CFS not operating, nor ESCAPE flag set
*****

 F995    PHP             ;save flags on stack
 F996    BIT     &EB     ;CFS Active flag
 F998    BMI     &F99E   ;
 F99A    BIT     &FF     ;if ESCAPE condition
 F99C    BMI     &F9A0   ;goto F9A0
 F99E    PLP             ;get back flags
 F99F    RTS             ;return
         ;


 F9A0    JSR     &F33B   ;close input file
 F9A3    JSR     &FAF2   ;enable second processor and reset serial system
```

```
F9A6    LDA     #&7E    ;A=&7E (126) Acknowledge ESCAPE
F9A8    JSR     OSBYTE  ;OSBYTE Call

F9AB    BRK             ;
F9AC    DB      &11     ;error 17
F9AD    DB      'Escape' ;
F9B3    BRK             ;
```

```
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 JIT 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