```
*************************************************************************
*                                                                       *
*                                                                       *
*       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   ;
```