GoDot

Commented Source
Source Format: MxAss (like Hypra-Ass)

```
.petscii
; current version is 1.28

; History:
;
; 24.04.97, Kernel Version 1.27
;
; 27.10.00, complete text revision
; 06.04.04, sprite beyond left border bug fixed
;

        .ob "godmain,p,w"
        .ba $1000

; ------------------------------------------------- Table of Box Types

gd_boxtab   .by 125,  95,  32,120,  32, 98,  32,123,  92          ; 0
            .by 106,118,116,101,  32,100,  99,113,102             ; 1
            .by  97,117,116,111,  32,119,  99,115,107             ; 2
            .by 110,117,100,111,  32,119,  99,115,107             ; 3
            .by 125,  95,100,120,  32,119,113,115,107             ; 4
            .by 125,  95,  32,120,  32,  98,113,115,113           ; 5
            .by 110,  95,  32,111 , 32,  98,  99,115,114          ; 6
            .by 104,117,116,120,  32,119,  32,123,108             ; 7
            .by 125,  95,100,120,  32,119,  32,123,108            ; 8
            .by  97,117,116,111,  32,119,101,123,108              ; 9
            .by 110,  95,100,111,  32,119,101,123,108             ; 10
            .by  97,117,116,111,  32,119,110,123,108              ; 11
            .by 104,117,118,120,  32,  98,  32,123,  92           ; 12
            .by  97,117,118,111,  32,  98,101,123,  92            ; 13
            .by  97,117,118,111,  32,  98,110,123,  92            ; 14
            .by 110,  95,  32,111,  32,  98,101,123,  92          ; 15

; one entry consists of 9 chars that define the borders and the inner of a box (space=32)
; table gets read from last to first byte (last is upper left corner)

; ------------------------------------------------
; Equations
; ------------------------------------------------

        .eq pport           =$01        ; usually $36 (BASIC off)
        .eq gd_oldposx      =$1d        ; mouse pointer
        .eq gd_oldposy      =$1e        ; dto
        .eq ls_loadsave     =$1f        ; flag for disk activities
        .eq sc_vekt20 =$20          ; usually screen address vector
        .eq sc_zl           =$22        ; row (in chars)
        .eq sc_sp           =sc_zl+1    ; column
        .eq sc_br           =sc_sp+1    ; width
        .eq sc_ho           =sc_br+1    ; height
```

```
.eq ls_lines    =$26                ; # of displayed lines in dir file window
.eq ls_found    =$28                ; # of matching files per dir block

.eq gd_soffx    =$7a                ; offset x of mouse pointer hotspot
.eq gd_soffy    =$7b                ; offset y

.eq status              =$90        ; status of IO device
.eq sc_pos              =$a4        ; various purposes
.eq sc_loop             =$a6        ; flag: text/color ram
.eq ls_temp             =$a7        ; various p.
.eq ls_vekta8           =$a8        ; usually a 16 bit counter
.eq gr_nr               =$aa        ; # of kernel function executed by upmem

.eq sc_screentab        =$ab        ; vector into screenlist
.eq gd_modswitch        =$ad        ; module switch management
.eq sc_texttab          =$ae        ; vector to text to be written on screen
.eq sc_merk             =$b0        ; various p.
.eq ls_adrmisc          =$b2        ; dto
.eq sc_flags            =$b4        ; flag: display type (in screenlist)
.eq lfname              =$b7        ; length of filename
.eq pfname              =$bb        ; pointer to filename
.eq sc_taste            =$c6        ; # of keys pressed
.eq ls_nambuf=$0200         ; filenames etc.
.eq gd_tbuffer          =$0277      ; key buffer
.eq video               =$0288      ; video mode
.eq sc_shadow           =$0334      ; color: shadow
.eq sc_light            =sc_shadow+1; color: light
.eq ls_showfiles        =$0339      ; flag: show all files (if >0)
.eq combuf              =$033a      ; buffer for REU command
.eq ls_lastname         =$03dc      ; most recent file in memory
.eq swapw               =$03ec      ; record for REU access (command parameters)
.eq ls_sysdrive         =$03fa      ; current system drive
.eq rm_ramvec           =$03fd      ; GoDot entry point for REU access routines

.eq gr_qtab             =$0ee0      ; quantization table (16 bytes)
.eq sc_screenvek        =$0f88      ; vector to screen output events
.eq gd_event            =$0f8a      ; vector to mouse click event routine
.eq gd_irqalt           =$0f8c      ; buffer for old IRQ (unused)
.eq sc_maincolor        =$0f8e      ; text screen background and border color

.eq gd_newposx          =$0f91      ; mouse position
.eq gd_newposy          =gd_newposx+1       ; dto
.eq sc_scvek2 =gd_newposy+1         ; backup of sc_screenvek
.eq gr_cmode =$0f95                 ; current graphics mode
.eq gr_howmany          =gr_cmode+1 ; # of greys to be rendered
.eq gr_picked =gr_cmode+2           ; current color (in palette)
.eq ls_drive            =$0fa8      ; current drive
.eq ls_dirmask          =ls_drive+1 ; file access type flag
.eq devs                =ls_dirmask-7 ; list of drives (8 - 11) and their types
.eq ls_track            =$0faf      ; current track
.eq ls_sector           =ls_track+1 ; current sector
.eq ls_index            =ls_track+2 ; vector into current dir block
.eq ls_err1             =ls_track+5 ; 1st digit of floppy error message (unused)
.eq ls_err2             =ls_track+6 ; 2nd digit
```

```
.eq ls_cblocks=ls_track+7          ; # of blocks containing matching files
.eq sc_clicked=ls_track+8          ; doubleclick counter
.eq sc_ticks        =ls_track+9    ; preload value of "sc_clicked"
.eq ls_flen         =ls_track+10   ; file length
.eq ls_flen2        =ls_track+11   ; file length minus 4 (without file signature)
.eq ls_first        =ls_track+12   ; first sector in directory
.eq gr_dither       =ls_track+13   ; dither type
.eq gr_redisp       =ls_track+14   ; flag: re-render graphics
.eq ls_saveto       =$0fc3         ; drive to save to
.eq ls_loadfrom     =ls_saveto+1   ; drive to load from
.eq sc_stop         =ls_saveto+2   ; flag: STOP was pressed
.eq sc_movetab      =$0fd8         ; buffer for texts and colors (39 bytes)

.eq blocks          =$3f40
.eq ftrack          =$3fa0

.eq intout          =$bdcd         ; C64 Kernal Routine "convert to int"

.eq dirbuf          =$bf00         ; directory buffer (256 bytes)
.eq modul           =$c000         ; Execution Area (where active module resides)
.eq forceoff        =$cb08         ; jump into dev.REU

.eq spritex         =$d002         ; mouse pointer parameters
.eq spritey         =spritex+1
.eq spritehi        =$d010

.eq potx            =$d419         ; mouse parameters
.eq poty            =potx+1
.eq joy       =$dc00

.eq reubase         =$df00         ; REU registers
.eq reucom          =reubase+1

.eq irqend          =$ea7e         ; C64 Kernal Vectors
.eq reset           =$fce2
.eq second          =$ff93
.eq tksa            =$ff96
.eq checktast =$ff9f
.eq iecin           =$ffa5
.eq ciout           =$ffa8
.eq untlk           =$ffab
.eq unlsn           =$ffae
.eq listen          =$ffb1
.eq talk            =$ffb4
.eq filpar          =$ffba
.eq filnam          =$ffbd
.eq copen           =$ffc0
.eq cclose          =$ffc3
.eq chkin           =$ffc6
.eq ckout           =$ffc9
.eq clrch           =$ffcc
.eq basin           =$ffcf
.eq basout          =$ffd2
```

```
; ------------------------------------------------ Compute Screen Address

initmove    ldy #0
            sty sc_vekt20
            sty sc_pos+1
            lda sc_loop                 ; flag: text ram or color ram
            bne im1
            lda video
            .by $2c
im1         lda #$d8
            sta sc_vekt20+1
            lda sc_zl
            sta sc_pos
            jsr mal40
            clc
            lda sc_pos
            adc sc_vekt20
            sta sc_pos
            lda sc_pos+1
            adc sc_vekt20+1
            sta sc_pos+1
            clc
            lda sc_pos
            adc sc_sp
            sta sc_vekt20
            lda sc_pos+1
            bne pl1
;
mal40       asl sc_pos                  ; multiply .a by 40
            rol sc_pos+1
mal20       asl sc_pos
            rol sc_pos+1
mal10       asl sc_pos
            rol sc_pos+1
            lda sc_pos+1
            pha
            lda sc_pos
            pha
            asl sc_pos
            rol sc_pos+1
            asl sc_pos
            rol sc_pos+1
            clc
            pla
            adc sc_pos
            sta sc_pos
            pla
            adc sc_pos+1
            sta sc_pos+1
            rts
;
plus40      clc                         ; add 40 to sc_vekt20
            lda sc_vekt20
            adc #40
            sta sc_vekt20
```

```
                lda sc_vekt20+1
pl1             adc #0
                sta sc_vekt20+1
                rts

; ------------------------------------------------ Move buffer

backu           ldy #0
bk2             lda sc_movetab,y
                sta (sc_vekt20),y
                iny
                cpy sc_br
                bne bk2
                rts

; ------------------------------------------------ Fill buffer

blank           dey
                lda #32                 ; ...with spaces
bl3             sta sc_movetab,y        ; ...with value in .a
bl4             dey
                bpl bl3
                rts

; ------------------------------------------------ Fill screen area by color

fcol            inc sc_loop             ; care for colorram
                jsr initmove            ; compute address
                lda sc_shadow,x         ; get color value (indexed by .x)
fi0             ldy sc_br               ; entry: fill by char (e.g. space)
                jsr bl4
                ldx sc_ho
fi1             jsr backu               ; entry: write buffer .x times
                jsr plus40
fi2             dex
                bne fi1
                rts

; ------------------------------------------------ Invert screen area

invert          ldx #0
inv0            ldy #0
inv1            cpy sc_br
                beq inv2
                lda (sc_vekt20),y
                eor #$80
                sta (sc_vekt20),y
                iny
                bne inv1
inv2            inx
                cpx sc_ho
                beq ready
                jsr plus40
                bne inv0
ready           rts
```

```
; ------------------------------------------------ Display box

box         jsr initmove            ; compute screenaddress
            stx gr_nr               ; compute offset in boxtype list (# times 9)
            txa
            clc
            asl
            asl
            asl
            adc gr_nr
            sta ls_vekta8
            lda #>(gd_boxtab)
            sta ls_vekta8+1
            ldy #8                  ; offset to last char of definition string
            sty sc_merk
            jsr makeline            ; create upper edge
            jsr backu               ; display
            jsr plus40              ; next row
            jsr makeline            ; create middle line
            ldx sc_ho
            dex
            jsr fi2                 ; display sc_ho times
            jsr makeline            ; create bottom edge
            jsr backu               ; display

; ------------------------------------------------ Add colors to box

farbe       lda sc_flags            ; a pressed button gadget?
            beq m2
            jsr swapcol             ; yes, exchange colors
m2          ldx #0                  ; offset to color of shadow
            jsr fcol                ; colorize (left and upper edge)
            lda sc_br               ; save width...
            pha
            inx                     ; ...and set to 1
            stx sc_br
            lda gr_nr               ; care for upper right and lower left corners
            cmp #12
            bcc m4
            dec sc_ho
m4          jsr fcol                ; colorize right edge
            pla                     ; restore width
            sta sc_br
            inx                     ; set height to 1
            stx sc_ho
            lda gr_nr               ; care for upper right and lower left corners
            cmp #9
            bcc m5
            dec sc_br
m5          jsr fcol                ; colorize bottom edge
            lda sc_flags            ; was a pressed button gadget?
            beq m6                  ; no, finished
;
swapcol     ldx sc_shadow           ; yes, re-exchange colors
```

```
                lda sc_light
                stx sc_light
                sta sc_shadow
m6              rts

; ---------------------------------------------- Create 1 Line of a box

makeline        ldy sc_merk             ; offset in definition string
                lda (ls_vekta8),y       ; get left corner/edge
                tax
                dey
                lda (ls_vekta8),y       ; get middle
                pha
                dey
                lda (ls_vekta8),y       ; get right corner/edge
                dey
                sty sc_merk             ; save current offset (for next line)
                ldy sc_br               ; set right corner/edge
                dey
                sta sc_movetab,y
                dey
                pla                     ; set middle
                jsr bl3
                stx sc_movetab          ; set left corner/edge
                rts

; ---------------------------------------------- Get screen position parameters

setpos          lda (sc_screentab),y    ; ...from screenlist
                sta sc_zl,x
                iny
                inx
                rts

; ---------------------------------------------- Add .Y to sc_screentab

setvektor       tya                     ; move vector into screenlist
                clc
                adc sc_screentab
                sta sc_screentab
                lda sc_screentab+1
                adc #0
                sta sc_screentab+1
                rts

; ---------------------------------------------- Display a Screenlist

screen          stx sc_screentab        ; pointer to list in .x/.y
                stx sc_screenvek
                sty sc_screentab+1
                sty sc_screenvek+1
                ldy #0
                lda (sc_screentab),y
                beq sc0
                jsr basout
```

```
sc0         iny
            bne sc1
;
s1          ldy #0
            ldx #0
            sty sc_loop
            sty sc_flags
s0          jsr setpos
            cpx #4
            bne s0
            lda (sc_screentab),y
            pha
            and #15
            tax
            pla
            pha
            and #$20
            sta sc_flags
m0          iny
            iny
            tya
            pha
            jsr box
            pla
            tay
            pla
            bpl m1
            jsr text                ; display additional texts
m1          iny
            lda (sc_screentab),y
            bmi text0
sc1         jsr setvektor
            bne s1
ready1      rts
;
text0       asl                     ; End of List? ($80)
            bpl ready1              ; yes, finished
            jsr text2
            bne m1

text2       iny
            jsr setvektor
text1       ldx #0
            ldy #0
t5          jsr setpos
            cpx #3
            bne t5
            lda #$10
            dey

; ---------------------------------------------- Output Text

text        and #16                 ; highlighted?
            sta sc_flags
            ldx #$00                ; care for text screen
```

```
                stx sc_loop
                dex
t1              inx                     ; output
                iny
                lda (sc_screentab),y
                sta sc_movetab,x
                bne t1                  ; until $00
                stx ls_temp             ; actual length
                sec                     ; compute indent for centering
                lda sc_br
                sbc ls_temp
                lsr
                stx sc_br
                tax
                dex
t2              inc sc_sp               ; indent text
                dex
                bpl t2
                inc sc_zl
                ldx #1
                stx sc_ho
                tya
                pha
                jsr initmove            ; compute address
                jsr backu               ; display
                inx
                lda sc_flags            ; hilite?
                beq t3
                inx                     ; yes, colorize
t3              jsr fcol
                pla
                tay
                rts

; ------------------------------------------------ Calculate Sprite position from Screen position

xy2spzl         sec                     ; subtract offset of hotspot
                sbc sc_merk+1
                lsr                     ; divide by 8
                lsr
                lsr
                rts

; ------------------------------------------------ Compute position of mouse pointer

position        ldx gd_soffy            ; convert sprite position to screen position
                stx sc_merk+1           ; hotspot y
                lda spritey
                jsr xy2spzl
                sta sc_merk
                ldx gd_soffx            ; hotspot x
                stx sc_merk+1
                lda spritex
                jsr logo
                jsr xy2spzl
```

```
                ldx spritehi
                beq po1
;               cmp #29                 ; care for upper sprite area
;               bcs po1
                clc
                adc #32
po1             sta sc_merk+1
                rts

; ------------------------------------------------ Check for click event

evntsuch        stx sc_screentab        ; vector to screenlist in .x and .y
                sty sc_screentab+1
                jsr position            ; where's the pointer?
                ldy #0
                beq esm1
es1             ldx #0                  ; care for text screen
                stx sc_loop
                ldy #0
                lda (sc_screentab),y    ; End of List? ($c0 or $80)
                bmi esready             ; yes, finished
;
es0             jsr setpos              ; get gadget parameters
                cpx #4
                bne es0
                lda (sc_screentab),y
                pha
                asl
                bpl esm0
                jsr trim
                sec
                lda sc_merk
                sbc sc_zl
                bcc esm0
                tax
                cpx sc_ho
                bcs esm0
                sec
                lda sc_merk+1
                sbc sc_sp
                bcc esm0
                tax
                cpx sc_br
                bcs esm0
                ldy #5
                lda (sc_screentab),y
                sta gd_event
                iny
                lda (sc_screentab),y
                sta gd_event+1
                pla
                jsr initmove
                jsr invert
                ldx #$40
                jsr dl2
```

```
esm3        jsr initmove
            jmp invert
;
esm0        ldy #6
            pla
            bpl esm1
esm2        iny
            lda (sc_screentab),y
            bne esm2
esm1        iny
            jsr setvektor
            bne es1

; ---------------------------------------------- Simple Delay

delay       ldx #4
dl2         ldy #$c7
dloop       dey
            bne dloop
            dex
            bne dloop
esready     rts

; ---------------------------------------------- Change params to gadget's inner area

trim        inc sc_zl
            inc sc_sp
            dec sc_br
            dec sc_br
            dec sc_ho
            dec sc_ho
            rts

; ---------------------------------------------- IRQ (controls all interface devices)

gd_irq      lda potx                    ; mouse
            ldy gd_oldposx
            jsr moved
            sty gd_oldposx
            clc
            adc spritex
            sta spritex
            sta spritex-2
            txa
            adc #0
            and #1
            bne y0
            lda #0
            .by $2c
y0          lda #$03
            eor spritehi
            sta spritehi
;
            lda poty
            ldy gd_oldposy
```

```
            jsr moved
            sty gd_oldposy
            sec
            eor #$ff
            adc spritey
            sta spritey
            tax
            dex
            dex
            stx spritey-2
;
tastatur    jsr checktast           ; keyboard: use ROM via $028f/90
            lda sc_taste            ; no key
            beq stick
            lda gd_tbuffer-1,x      ; get key
            cmp #3                  ; STOP key?
            bne tt
            sta sc_stop             ; yes, set flag
            beq ttt
tt          cmp #$0d                ; RETURN?
            beq stick
            cmp #$a0                ; Shift Space? (Left Mouse button)
            beq stick
ttt         ldx #0                  ; clear keyboard
            stx sc_taste
            cmp #$11                ; CRSR right?
            beq st10
            cmp #$91                ; CRSR left?
            beq st09
            cmp #$1d                ; CRSR down?
            beq st12
            cmp #$9d                ; CRSR up?
            beq st13
;
stick       lda joy                 ; joystick:
            and #$10                ; Fire?
            bne st0
            inc sc_taste
st0         lda joy                 ; right?
            and #1
            bne st1
st09        lda #$fc
            bne st11
st1         lda joy                 ; down?
            and #2
            bne st2
st10        lda #4
st11        clc
            adc gd_oldposy
            sta gd_oldposy
st2         lda joy                 ; left?
            and #4
            bne st3
st13        lda #4
            bne st31
```

```
st3         lda joy                 ; up?
            and #8
            bne st4
st12        lda #$fc
st31        clc
            adc gd_oldposx
            sta gd_oldposx
st4         lda sc_clicked          ; doubleclick active?
            beq st5
            dec sc_clicked          ; yes, decrease
st5         lda $dc0d               ; finish IRQ
gd_endirq   pla
            tay
            pla
            tax
            pla
gd_endnmi    rti

; ---------------------------------------------- Subroutines Mouse

moved       sty gd_newposy
            sta gd_newposx
            ldx #0
            sec
            sbc gd_newposy
            and #$7f
            cmp #$40
            bcs more
            lsr
            beq most
            ldy gd_newposx
            rts
;
more        ora #$c0
            cmp #$ff
            beq most
            sec
            ror
            ldx #$ff
            ldy gd_newposx
            rts
;
most        lda #0
            rts
```

```
; ---------------------------------------------- Version Number

gd_version  .by $01,$28             ; SED-mode

; ----------------------------------------------
```

```
; ------------------------------------------------
; Entry to GoDot, Main Loop
; ------------------------------------------------

into        ldx #<(mainlst)         ; main loop, pointer to list
            ldy #>(mainlst)
;
main        jsr mloop               ; display screenlist and wait for events
            bne into

; ------------------------------------------------
; ------------------------------------------------
; ------------------------------------------------

gd_xmloop   jmp mloop

; ------------------------------------------------ Event Loop

gd_eloop    lda #0                  ; wait for events
            sta sc_taste            ; clear keyboard
el0         lda sc_taste            ; no change...
            beq el0                 ; ...so wait
            ldx sc_screenvek        ; vector to current screenlist
            ldy sc_screenvek+1
            jsr evntsuch            ; check
            bmi gd_eloop            ; no valid click, wait
            jsr exec                ; valid click, execute command
            bcc gd_eloop            ; finished event, but stay in list
            rts                     ; definitely finished, back to main screen
;
exec        jmp (gd_event)          ; execute function the user clicked on

; ------------------------------------------------ backup sc_screenvek

savescvek   lda sc_screenvek
            sta sc_scvek2
            lda sc_screenvek+1
            sta sc_scvek2+1
            rts

; ------------------------------------------------ Event: Exit

ev_exit     ldx #<(exlst)           ; display requester
            ldy #>(exlst)
            jsr mloop               ; wait

; ------------------------------------------------ Subevent: Cancel

se_cancel   sec                     ; continue in main
            rts

; ------------------------------------------------ Subevent: really Exit

se_ende     jmp reset               ; finish GoDot
```

```
; ---------------------------------------------
;Table of Vectors
; ---------------------------------------------

gd_xopen    jmp fopen               ; open a file
gd_xclose   jmp gd1                 ; close a file
gd_xmess    jmp sevch15             ; display floppy error message
gd_xtxout1  jmp scc5                ; text out
gd_xtxout2  jmp error2              ; text out
gd_xtxout3  jmp out2                ; text out
gd_xinput   jmp se_input            ; input from keyboard
gd_xtxtggl  jmp tggl2               ; toggle
gd_xswap4k  jmp swap4k              ; exchange modules
gd_xcnvdez  jmp cnvdez              ; convert to decimal
gd_xloadm   jmp lsm0                ; load module

; ---------------------------------------------

magic       .by $ad

; ---------------------------------------------
;IO routines
; ---------------------------------------------

gd_listen   jsr listen
            tya
            jmp second
;
gd_talk     jsr talk
            tya
            jmp tksa
;
; ---------------------------------------------- Send IO command

gd_sendcom jsr filnam               ; send IO command
            lda ls_drive
scm1        ldy #$6f
            jsr gd_listen
            ldy #0
scm0        lda (pfname),y
            jsr ciout
            iny
            cpy lfname
            bcc scm0
            jmp unlsn

; ---------------------------------------------- Check for presence of drive (# in X)

present     lda #0
            sta status
            sta ls_drive
            txa
            ldy #$6f
            jsr gd_listen
            lda status
```

```
              bmi pre0
              txa
              sta ls_drive
pre0          jsr setdrive
              jmp unlsn

; ---------------------------------------------- set presence flag

setdrive      lda devs,x
              and #$f0
              ora ls_drive
              sta devs,x
td            rts

; ---------------------------------------------- Check current drive for presence

gd_testdrive  jsr gd_testram
              bcs td
              tax
              jsr present
              lda ls_drive
              bne tdok
              inc ls_err2
              bne error1

; ---------------------------------------------- Set parameters for GoDot's Statusbar

gd_setmess    lda #23                 ; row
              ldx #4                  ; column
              ldy #32                 ; width
gd_setpar     sta sc_zl               ; external entry: set their own parameters
              stx sc_sp
              sty sc_br
              ldx #1                  ; height
              stx sc_ho
              dex
              stx sc_loop             ; care for text screen
              rts

; ---------------------------------------------- Clear Statusbar

gd_clrms      jsr gd_setmess          ; focus to statusbar
gd_clrline    jsr blank               ; create blank message
              jsr initmove            ; compute screen address
              jsr backu               ; output the message
sdw3          clc
              rts

; ---------------------------------------------- Message: "Drive off."

error1        ldx #<(nodrive)
              ldy #>(nodrive)

; ---------------------------------------------- Display Message
```

```
error2      stx sc_screentab
            sty sc_screentab+1
            jmp scc4

; ------------------------------------------------ Check all drives for presence

which       lda ls_drive            ; save current drive
            pha
            ldx #8                  ; check, beginning with drive #8
whi0        jsr present             ; if present, report to GoDot
            inx
            cpx #$0c                ; end at drive #11
            bne whi0
            pla
            sta ls_drive            ; restore current drive
            rts

; ------------------------------------------------

tdok        sec
            rts

; ------------------------------------------------ Subevents: Scroll Dir

se_scdwn    lda ls_cblocks          ; scroll ahead, get # of dirblocks in window
            bne sdw4
            lda #2                  ; if none, set to 2
            .by $2c

se_scup     lda #0                  ; scroll back
sdw4        pha
            lda #1                  ; clear file window
            jsr sun2
            pla
            tay
            iny
sdw5        jsr gd_testram          ; RAM active?
            bne sdw6
            ldx #5                  ; yes, RAM dir (function #5)
            jmp rm_ramvec
sdw6        ldx blocks              ; index into list of dirblocks
            dex
            bne sdw2
            ldy ls_first            ; if 0: get first block
            bne sdw10               ; uncond. jump

sdw0        dex
            bne sdw2
            inx
sdw2        dey                     ; I know this must mean something...
            bne sdw0

sdw1        ldy blocks,x            ; get sector of dirblock
            stx blocks
sdw10       lda ftrack,x            ; get track of dirblock
```

```
                sta ls_track
                ldx #1                  ; start display of filenames in line #1
                stx sc_zl
                jmp dir4                ; display

; ---------------------------------------------- Open Dir window

prepdir         lda #2                  ; draw two boxes
sun2            ldy #3                  ; external entry: draw file window (.a=1 for one)
                sta sc_merk+1
sun1            ldx #3                  ; set parameters
sun0            lda dirwind,y
                sta sc_zl,x
                dey
                dex
                bpl sun0
                inx                     ; care for text screen
                stx sc_loop
                lda sc_merk             ; save value (# of window entry)
                pha
                lda #$20                ; draw box
                sta sc_flags
                jsr box
                pla                     ; restore value
                sta sc_merk
                ldy #10                 ; offset to second box
                dec sc_merk+1           ; second box?
                bne sun1                ; yes, loop
                rts

; ---------------------------------------------- Activate Doubleclick

clickon         lda sc_ticks            ; get speedvalue
                sta sc_clicked          ; set
                rts

; ---------------------------------------------- Subevent: Select a File

se_select       lda sc_clicked          ; doubleclick?
                beq sel3
                jmp se_load             ; yes, load file

sel3            jsr clickon             ; no, activate doubleclick counter
                ldx #1                  ; clear filename box (below file window)
                stx sc_ho
                ldx #$14
                stx sc_zl
                ldy sc_br
                jsr gd_clrline
                lda sc_merk             ; get # of file window entry
                sta sc_zl
                jsr initmove            ; compute screen address
                ldx ls_dirmask          ; filetype?
                beq sel0                ; 0: graphics file
```

```
                ldx #4                  ; not 0: a module (no prefix, indent 4 chars)
sel0            lda (sc_vekt20),y       ; get filename (from screen!)
                sta sc_movetab,y        ; store to buffer for screen output
                sta ls_lastname,y       ; store to buffer of current filename
                sta ls_nambuf,x         ; store to buffer for open disk procedure
                inx
                iny
                cpy sc_br
                bne sel0                ; finished when width of box equalled

sel1            dey                     ; strip trailing spaces
                lda sc_movetab,y
                cmp #$20
                beq sel1

                iny                     ; length of name
                bne sel4
                sty sc_clicked          ; if 0: stop doubleclick counter
                beq cva4                ; finished

sel4            sty sc_br               ; save (needed for centering)
                sty ls_flen2
                lda #0                  ; close string
                sta sc_movetab,y
                lda #20                 ; output always on row #20
                sta sc_zl
                jsr out2
                lda ls_dirmask          ; graphics file?
                beq cnvasc
cnvafil         ldy #0                  ; no, add system prefix
sel5            lda filtype,y
                sta ls_nambuf,y
                inc sc_br
                iny
                cpy #4
                bcc sel5
                beq cva3                ; and convert name to (PET)ASCII

; ------------------------------------------------ Convert Screencode to PETSCII

cnvasc          ldy #0
cva3            lda ls_nambuf,y         ; end of string?
                beq cva4                ; yes, finished
                sta ls_temp
                and #$3f
                asl ls_temp
                bit ls_temp
                bpl cva0
                ora #$80
cva0            bvs cva1
                ora #$40
cva1            sta ls_nambuf,y
                iny
                cpy sc_br
                bne cva3
```

```
cva4        sty ls_flen             ; set length of filename
urd0        clc
            rts

; ------------------------------------------------ Subevent: Select Drive

se_units    lda #1                  ; clear file window (not: filename box)
            jsr sun2
            jsr units               ; display currently active units
suni        ldx #1                  ; external entry: directory starting with line #1
            stx sc_zl
            jmp gd_dir
;
units       jsr uready              ; clear marker on previous drive
            lda sc_merk             ; drive number is line# plus 5
            adc #5
            tax
            cpx ls_drive            ; new drive = previous drive?
            beq urd3                ; yes, finished

urd1        lda ls_drive            ; save prv drive
            pha
            txa                     ; set new drive
            sta ls_drive
            jsr gd_clrms
            jsr gd_testdrive        ; test drive presence
            pla                     ; (get prv drive)
            bcs uready              ; yes, present
urd4        sta ls_drive            ; no, re-set prv drive

uready      lda ls_drive            ; show marker on current drive
            sec                     ; get parameters
            sbc #5
            sta sc_zl
            lda #11
            sta sc_br
            lda #24
            sta sc_sp
            ldy #1
            sty sc_ho
            dey                     ; care for textscreen
            sty sc_loop
            jsr esm3                ; invert area (leaves with .x=sc_ho=1)

urd7        jsr gd_testram          ; RAM active?
            bne urd0                ; no, finished

            lda modul               ; RAM: correctly switched in?
            adc sc_merk
            cmp #8
urd2        beq urd0                ; yes, finished

            cmp #$18                ; invoked from "Execute/Delete"?
            bcc urd6                ; yes, enter ram device, function "Init" (.x=1)
```

```
                jsr gd_swapd            ; no, RAM disabled, switch on again
                inx                     ; now .x=0
urd6            jmp rm_ramvec           ; enter RAM device
;
urd3            jsr gd_clrms            ; clear status bar
urd5            jsr gd_testdrive        ; drive still active?
                bcs uready              ; yes, switch marker on again
                lda #12                 ; no, change current drive to RAM (drive #12)
                bne urd4

; ---------------------------------------------- Subevent: Read Error Channel

sevch15         jsr gd_testram          ; RAM active?
                beq urd0                ; yes, finished

read15          jsr gd_clrms            ; clear status bar
                jsr gd_testdrive        ; drive still active?
                bcc urd0                ; no, finished

rd12            ldx #0                  ; retrieve error message
                stx status
rd11            lda ls_drive
                ldy #$6f
                jsr gd_talk
rd10            jsr iecin               ; read bytes
                sta sc_movetab,x        ; and store them
                inx
                eor #$0d                ; until RETURN
                bne rd10

                sta sc_movetab-1,x      ; close string by 0
                jsr untlk               ; close error channel

                lda sc_movetab          ; store error code, 1st digit
                sta ls_err1
                beq rd12                ; if failure: try again
                nop
                lda sc_movetab+1        ; store 2nd digit
                sta ls_err2
out2            ldx #<(sc_movetab)      ; output message (main output entry)
                ldy #>(sc_movetab)
                jmp error2

; ---------------------------------------------- Subevent: Delete File

se_delete       jsr gd_testram          ; RAM active?
                bne dlt1
                ldx #4                  ; yes, function #4, Exec
                jmp rm_ramvec

dlt1            lda ls_flen             ; any file selected?
                beq urd2                ; no, finished

                tay                     ; move filename 2 digits to the right
                tax
```

```
                iny
                iny
                sty lfname
dlt0            lda ls_nambuf-1,x
                sta ls_nambuf-1,y
                dey
                dex
                bne dlt0
                lda #$53                ; add "s:" at string start
                sta ls_nambuf
                lda #$3a
                sta ls_nambuf+1
                jsr gd_sproff           ; switch Sprites off
                ldy #2                  ; send command
                lda lfname
                jsr gd_sendcom
                jsr sevch15             ; error message (answer from drive)
                stx ls_flen             ; .x=0
                jsr prepdir             ; display changed dir
                dec ls_cblocks
                jmp se_scdwn

; ------------------------------------------------ Exchange Modules

swap4k          lda #$c0                ; source area $c000
swappfox        sta sc_texttab+1
                txa                     ; hi-byte of target area in .x
                sta sc_pos+1
                ldy #0
                sty sc_texttab
                sty sc_pos
;
                sei                     ; switch C64 to RAM throughout
                lda 1
                pha
                lda #$30
                sta 1
                ldx #16                 ; 16 pages (=4096 bytes)
swp0            lda (sc_texttab),y      ; get source
                pha
                lda (sc_pos),y          ; get target
                sta (sc_texttab),y      ; store each
                pla
                sta (sc_pos),y
                iny
                bne swp0
                inc sc_texttab+1
                inc sc_pos+1
                dex
                bne swp0                ; until finished
                pla
                sta 1
                cli
                rts
```

```
; ---------------------------------------------- Event: Execute Module

ev_exec     lda #$e0                ; mods at $e000
            bne evb2

; ---------------------------------------------- Subevent: Save File

se_save     bne ss1                 ; determine whether RAM
            jmp ld11                ; RAM active, jump back
ss1         tya                     ; .y=0 (tya is dummy)
            .by $2c                 ; function number 0 (Save)

; ---------------------------------------------- Subevent: Input Text

se_input    ldy #1                  ; function # 1
            lda #$d0                ; savers at $d000
            bne evb1

; ---------------------------------------------- Event: Redisplay

ev_disp2    ldy #7                  ; func # 7
            .by $2c

; ---------------------------------------------- Event: Display Preview

ev_prviu    ldy #6                  ; func # 6
            .by $2c

; ---------------------------------------------- Event: Show Guru

ev_info     ldy #5                  ; func # 5
            .by $2c

; ---------------------------------------------- Event: Toggle Clip/Full

ev_area     ldy #4                  ; func # 4
            .by $2c
; ---------------------------------------------- Event: Select Dither Type

ev_dith     ldy #3                  ; func # 3
            .by $2c

; ---------------------------------------------- Event: Display Graphics

ev_display  ldy #2                  ; func # 2
            .by $2c

; ---------------------------------------------- Event: Palette

ev_pal      ldy #1                  ; func # 1
            .by $2c

; ---------------------------------------------- Event: Balancing

ev_balance  ldy #0                  ; func # 0
```

```
evb0        lda #$f0                ; all these func # for upmem

; ------------------------------------------------

evb1        sty gr_nr               ; store func #
evb2        sta gd_modswitch        ; module management:
slot4       tax                     ; exchange modules
            jsr swap4k
            jsr modul               ; execute module at $c000
getback     php                     ; save Carry flag
            ldx gd_modswitch        ; re-exchange modules
            jsr swap4k
            plp                     ; restore Carry flag
            rts                     ; and back to main loop

; ------------------------------------------------ Event: Select Graphics Mode

ev_sccont   lda #<(scrmode)         ; Toggle text on button
            sta sc_texttab
            lda #>(scrmode)
            sta sc_texttab+1
            sta gr_redisp           ; force redisplay
            lda gr_cmode            ; previous mode Hires?
            bne scc0
            lda #2                  ; yes, change to Multi
            ldx #<(mult)
            ldy #>(mult)
            bne scc1
scc0        lda #0                  ; no, change to Hires
            ldx #<(hirs)
            ldy #>(hirs)
scc1        sta gr_cmode
;
tggl2       stx sc_screentab        ; address of new text
            sty sc_screentab+1
            ldy sc_br               ; first clear gadget
            jsr blank
            jsr initmove
            jsr backu
            jsr swaptxt             ; exchange texts

scc4        ldy #$ff                ; output new text
scc5        dec sc_zl
            dec sc_sp
            lda #0
            jsr text
cont        clc
            rts

; ------------------------------------------------ Exchange texts

swaptxt     ldy #$ff
scc3        iny
            lda (sc_screentab),y
            sta (sc_texttab),y
```

```
            bne scc3
            rts

; ------------------------------------------------ Create Quantization Table

makeqtab    lda #0
            ldx #1
            clc
qt0         adc gr_howmany          ; cont. add # of grays/colors to be displayed
            tay
            lsr                     ; integer devide by 16
            lsr
            lsr
            lsr
            sta gr_qtab,x           ; store result to table
            tya
            inx
            cpx #16
            bne qt0
            rts

; ------------------------------------------------ Event: Select # of colors/grays

ev_cols     ldy gr_howmany          ; get current number
            lda sc_merk+1           ; check screen column
            cmp #17                 ; >= 17?
            bcs co2
            dey                     ; no, decrease #
            cpy #2                  ; down to 2
            bcs co0
            ldy #16                 ; if lower, start over
            bne co0
co2         iny                     ; yes, increase #
            cpy #17                 ; up to 16
            bcc co0
            ldy #2                  ; if higher, start over
co0         tya
            sta gr_howmany          ; store new number
            sta gr_redisp           ; force redisplay
            jsr cnvdez              ; convert value to digits
            ldy #2                  ; enter digits in screenlist
            sta numcols,y
            txa
            and #$0f
            bne co1
            ldx #$20
co1         txa
            dey
            sta numcols,y
            jsr makeqtab            ; create new quantization table
            ldy #6                  ; display digits
            bne scc5

; ------------------------------------------------ Subevent: Load File
```

```
se_load     ldy ls_loadsave     ; IO mode?
            bne ld11            ; is load mode
            jsr gd_testram      ; no, is save mode, RAM active?
            jmp se_save         ; continue at SAVE

ld11        lda ls_flen         ; file selected?
            beq cont            ; no, finished
            sta ls_temp         ; store length of filename
            lda #2              ; name at $0200
            sta ls_vekta8+1
ld12        lda #$c0
            sta ls_index
            sta sc_vekt20+1
            ldy #0
            sty ls_vekta8
            sty sc_vekt20
            lda ls_dirmask      ; filetype (0,1,2,4,8)
            bne ld0
ld4         lda #$40            ; 0: graphics to $4000
            sta sc_vekt20+1
            bne ld2
ld0         lsr                 ; 1: loader to $c000
            bcs ld1
            lsr                 ; 2: saver to $d000
            bcc ld3
            lda #$10
            bne ld1
ld3         lsr                 ; 4: modifier to $e000
            bcs ld31
            ldx ls_adrmisc      ; 8: miscellaneous data elsewhere
            ldy ls_adrmisc+1
            stx sc_vekt20
            sty sc_vekt20+1
ld2         lda #0
            beq ld1

ld31        lda #$20            ; add 32 to ls_index (1 entry)
ld1         jsr bindex          ; result in .x
            stx gd_modswitch    ; module management
            jsr gd_testram      ; RAM active?
            bne ld8
            ldx #3              ; yes, function 3, Install
            jmp rm_ramvec

ld8          jsr swap4k         ; no, exchange modules
            sty sc_texttab      ; .y=0
            sty sc_texttab+1
            sty status
            ldx ls_dirmask      ; misc?
            cpx #8
            bcc ld81
            rts                 ; yes, finished

ld81        txa                 ; graphics file?
            bne ld5
```

```
                inx                     ; yes, force redisplay
                stx gr_redisp
ld9             jmp modul               ; continue in loader

ld5             jsr fopen               ; no, modifier
                jsr gd_onebyte          ; skip address
                bne lderr
                jsr gd_onebyte
                bne lderr
ld6             lda status              ; file doesn't exist or error or EOF?
                bne ld7                 ; yes, finished
                                        ; (BVS didn't work with some fast loader devices)
                jsr basin               ; get file data
                sta (sc_vekt20),y
                iny
                bne ld6
                inc sc_vekt20+1
                bne ld6
ld7             jsr lderr               ; close file

chgtxt          lda ls_dirmask          ; re-get filetype (to enter names in proper gadget)
                lsr
                bcc ct0
                ldx #<(intype)          ; 1: loader
                ldy #>(intype)
                bne ct4
ct0             lsr
                bcc ct1
                ldx #<(outtype)         ; 2: saver
                ldy #>(outtype)
                bne ct4
ct1             lsr
ct9             bcc se_ldcan
                ldx #<(efftype)         ; 4: modifier
                ldy #>(efftype)
ct4             stx sc_texttab
                sty sc_texttab+1
                lda #0                  ; clear gadget first
                tax
                tay
                lda #$20
ct5              sta (sc_texttab),y
                iny
                cpy #12
                bne ct5
                tya
                sec
                sbc ls_flen2
                lsr
                tay
ct6             lda ls_lastname,x       ; and write new name entry
                sta (sc_texttab),y
                inx
                iny
                cpx ls_flen2
```

```
              bcc ct6
              jsr sevch15               ; final error status

; ------------------------------------------------ Subevent: Cancel Load

se_ldcan      ldx #1                    ; RAM active/loaded from RAM?
              jsr urd7
              bcc can0
              jsr forceoff              ; yes, definitely switch RAM off
can0          sec                       ; back to main loop
              rts

; ------------------------------------------------ Error

lderr         jsr sevch15               ; display error status
              stx ls_flen               ; .x=0 here
              jsr gd1                   ; close file
              jsr getback               ; re-exchange modules
              jmp gd_spron              ; switch mouse pointer on

; ------------------------------------------------
;Open Filerequester
; ------------------------------------------------

; ------------------------------------------------ Check Type of Module

modreq        lda modul+3               ; is module a loader?
              and #$80
              beq mr0
ownreq        lda modul+4               ; yes, is OwnReq flag set?
              and #4
mr0           rts

; ------------------------------------------------ Event: Load File

ev_load       ldx #1
              jsr modreq
              beq esv1
              jmp modul                 ; continue in loader if own requester

; ------------------------------------------------ Event: Save File

ev_save       ldx #0
esv1          lda #<(joker)
              pha
              lda #>(joker)
              pha
              lda #0
              beq lsm1

; ------------------------------------------------ Event: Install Loader

ev_ldr        lda #<(ldr)
              pha
              lda #>(ldr)
```

```
                pha
                lda #1
                bne lsm0

; ------------------------------------------------ Event: Install Saver

ev_svr          lda #<(svr)
                pha
                lda #>(svr)
                pha
                lda #2
                bne lsm0

; ------------------------------------------------ Event: Install Modifier

ev_mod          lda #<(mod)
                pha
                lda #>(mod)
                pha
                lda #4

; ------------------------------------------------

lsm0            tax                     ; external entry: load miscellaneous data

; ------------------------------------------------

lsm1            sta ls_dirmask          ; store filetype
lsm2            stx ls_loadsave         ; store mode (save=0; load=1 or 8)
                ldx #<(filtype)         ; change filter (prefix in filename)
                ldy #>(filtype)
                stx sc_texttab
                sty sc_texttab+1
                pla
                sta sc_screentab+1
                pla
                sta sc_screentab
                jsr swaptxt
                ldx #<(reqtype)
                ldy #>(reqtype)
                stx sc_texttab
                sty sc_texttab+1
                lda ls_loadsave         ; "Load"/"Save" into command gadget
                beq fil0
                ldx #<(txtload)
                ldy #>(txtload)
                bne fil1
fil0            ldx #<(txtsave)
                ldy #>(txtsave)
fil1            stx sc_screentab
                sty sc_screentab+1
                jsr swaptxt
                jsr savescvek
                ldx #<(filelst)         ; display requester
                ldy #>(filelst)
```

```
            jsr screen
; ------------------------------------------------
            lda ls_sysdrive

            ldx ls_dirmask          ; graphics?
            bne fil4
            ldx ls_loadsave         ; yes, save?
            bne fil6
            lda ls_saveto           ; yes, change to related drive
            bne fil7
fil6        lda ls_loadfrom         ; otherwise select load drive
            beq fil3
fil7        pha                     ; show current name (graphics)
            lda #<(picname)
            sta sc_vekt20
            lda #>(picname)
            sta sc_vekt20+1
            lda #5
            sta sc_sp
            lda #16
            sta sc_br
            ldx #0
            ldy #0
            jsr sel0
            pla
fil4        sta ls_drive            ; set drive
fil3        jsr showdrv             ; display drive types
            lda #3
            ldx #24
            ldy #11
            jsr gd_setpar
decpt       lda #5
            sta sc_ho
            ldx #2                  ; colorize display
            jsr fcol
            stx sc_taste            ; clear keyboard (.x=0)
            jsr urd3                ; show drive marker
reqpt       jsr suni                ; clear file window and show files

            lda ls_dirmask          ; finished, if misc
            and #8
            beq fil2
            rts
;
fil2        jsr gd_eloop            ; waiting for click (in file requester)

            php                     ; save Carry
            ldx ls_drive            ; fix last picture drive as current picture drive
            lda ls_dirmask
            bne fil5
            lda ls_loadsave
            beq app2
            stx ls_loadfrom
            bne fil5
app2        stx ls_saveto
```

```
fil5        plp                 ; restore Carry
            rts

; ------------------------------------------------ Set Buffer Address

setmtab     lda #<(sc_movetab)  ; vector to textbuffer
            sta sc_texttab
            lda #>(sc_movetab)
            sta sc_texttab+1
            rts

; ------------------------------------------------

sdrive      lda drvtype,y       ; write drivetype to buffer
            sta (sc_texttab),y
            dey
            bpl sdrive
            rts
;
sram        ldy #3              ; RAMtype texts
sra0        lda ramtype,x
            sta (sc_texttab),y
            dex
            dey
            bpl sra0
            rts

; ------------------------------------------------

setspos     ldx #0              ; set parameters for Units gadget
            stx sc_loop
            inx
            stx sc_ho
            inx
            inx
            stx sc_zl
            inx
            stx sc_br
            ldx #30
            stx sc_sp
            rts

; ------------------------------------------------ Show Drives

showdrv     jsr which           ; check drives
            jsr setspos         ; set parameters for output
            ldy #8              ; starting from drive #8 (to #11)

looping     lda devs,y          ; drive present?
            sty sc_merk
            tax
            and #$0f
            beq weiter          ; no, next one

            jsr setmtab         ; set text buffer address
```

```
                jsr sdrive              ; set default drive (1571)

                txa                     ; re-get drive flag
                and #$f0                ; 0 =1570
                beq s1570
                asl                     ; 8 =1581
                bcs s1581
                asl                     ; 4 =1541
                bcs s1541
                bpl sout
                ldx #19                 ; 2 =CMD Native

sdr4            jsr sram                ; enter text
sout            jsr initmove            ; compute screen address
                jsr backu               ; display

weiter          inc sc_zl               ; next line
                ldy sc_merk             ; next drive
                iny                     ; up to #11
                cpy #12
                bcc looping

; ------------------------------------------------ Show RAM drive

cmdpt           bne sdr7                ; next is RAM (12)
                sty sc_merk

shram           ldx #3                  ; default is 1700
                jsr sram
                lda devs+12             ; get RAM infoflag
                lsr                     ; even number?
                bcc sdr3                ; yes, see below

                lsr                     ; 3 =Pagefox cartridge
                bcs spfox
                lsr                     ; 5 =1750
                bcs s1750
                ldx #7                  ; 9 =C64 (no RAM)
                .by $2c
spfox           ldx #15
                .by $2c
svdc            ldx #11
                bpl sdr4                ; output drive type

sdr3            lsr                     ; 2 =VDC
                bcs svdc
                lsr                     ; 4 =1764
                bcs s1764
                lsr                     ; 8 =1700
                bcs sout
sdr7            rts

; ------------------------------------------------

s1764           ldx #3                  ; modify text for display
```

```
                lda #$34
                sta sc_movetab,x
                lda #$36
                .by $2c
s1750           lda #$35
                .by $2c
s1581           lda #$38
                .by $2c
s1541           lda #$34
sdr5            ldx #2
sdr6            sta sc_movetab,x
                bne sout
;
s1570           ldx #3
                lda #$30
                bne sdr6

; ------------------------------------------------
magic5      .by $ad,$ad
; ------------------------------------------------

; ------------------------------------------------
; Dir Routines
; ------------------------------------------------
;
getdir          lda #1                  ; read dir block
                ldx #<(dirchan)
                ldy #>(dirchan)
                sta ls_temp
                stx ls_vekta8
                sty ls_vekta8+1
                jsr open
                jsr send
                ldx #$0d
                jsr chkin
                ldy #$00
gd0             jsr basin
                sta dirbuf,y
                iny
                bne gd0
gd1             jsr clrch               ; close file
                jmp close

; ------------------------------------------------

send            ldx #$0f                ; send command
                jsr ckout
                ldx #$00
l98eb           lda messu1,x
                jsr basout
                inx
                cpx #$08
                bcc l98eb
                lda #0
                ldx ls_track
```

```
                inc pport
                jsr intout
                lda #$20
                jsr basout
                txa
                ldx ls_sector
                jsr intout
                dec pport
                jsr clrch
                jmp sevch15

; ---------------------------------------------- Convert Value to Digit

cnvdez          ldx #$30                ; value in .a
                sec
l98ff           sbc #$0a                ; tenths in .x
                bcc l9906
                inx
                bcs l98ff
l9906           adc #$3a                ; oneths in .a
                rts

; ------------------------------------------------

close           lda #$0d                ; close 13 and 15
                jsr cclose
                lda #$0f
                jmp cclose

; ------------------------------------------------

open            lda #$0f                ; open 15 and 13
                tay
                ldx ls_drive
                jsr filpar
                lda #$00
                jsr op0
                lda #$0d
                tay
                ldx ls_drive
                jsr filpar
                lda ls_temp
                ldx ls_vekta8
                ldy ls_vekta8+1
op0             jsr filnam
                jmp copen

; ------------------------------------------------
;Directory
; ------------------------------------------------

gd_dir          jsr gd_testram          ; RAM active?
                bne dir0
                ldx #2                  ; yes, function 2, Directory
                jmp rm_ramvec
```

```
; ------------------------------------------------

ftabt       .by 18,18,18,18         ; table of first dir blocks
ftabs       .by 1,1,1,1             ; drive 8 thr 11

; ------------------------------------------------

dir0        tax                     ; drive number
            lda ftabt-8,x
            ldy ftabs-8,x
            sta ls_track            ; set parameters
            sta ftrack
            sty ls_first
            ldx #1                  ; block counters
            stx blocks
            sty blocks+1
dir4        dex
            stx ls_index            ; dir index (into block)
            stx ls_lines            ; # of lines in window
            stx ls_cblocks          ; # of blocks with matching files
            jsr gd_sproff           ; pointer off

            tya                     ; retrieve dir block
            tax
dir2        stx ls_sector
            lda sc_zl
            pha
            jsr getdir
            pla
            sta sc_zl
            lda ls_err2
            and #$0f
            bne dir3
            jsr gd_makedir          ; process data and display
            bne dir2
dir3        jsr gd_spron            ; pointer on again
            clc                     ; continue until window filled
            rts                     ; or last dir block processed

; ------------------------------------------------ Process Directory Data

gd_makedir  lda #0
            sta ls_found
            lda #2
            jsr bindex
            lda dirbuf,x            ; get C= filetype
            and #7                  ; 0/5 =DEL/CBM; 6 =DIR
            beq mdr2
            cmp #5
            bcs mdr2
;
            lda #$10                ; length of filename (16)
            sta sc_merk+1
;
```

```
                lda ls_dirmask          ; filetype graphics?
                tay
                ora ls_showfiles        ; forced to show all?
                bne mdr8

                lda #8                  ; no, graph: filter all system files
                jsr bindex
                lda dirbuf,x
                cmp #$2e                ; having a dot as their 4th char
                beq mdr2
mdr8            ldx #$05
                tya
                beq ausgabe             ; display (if graphics)

; ---------------------------------------------

                lda #$0c                ; filter particular filetypes
                sta sc_merk+1           ; length 12 (16 minus prefix)
                lda #8
                jsr bindex
                lda dirbuf,x            ; dot on 4?
                cmp #$2e
                bne mdr2
;
                ldy #$03                ; compare prefixes
mdr0            dex
                lda dirbuf,x
                cmp filtype-1,y
                bne mdr1
                dey
                bne mdr0
mdr1            tya                     ; match if .y=0
                bne mdr2

; ---------------------------------------------

                ldx #$09                ; display filenames
ausgabe         stx sc_merk
                jsr out1
                jsr bin2
                ldy #$00
mdr3            lda dirbuf,x            ; name to text buffer
                sta sc_movetab,y
                inx
                iny
                cpy sc_merk+1
                bne mdr3
                jsr gd_cnvbc            ; convert to screen code
                tya
                beq mdr7
                sty sc_br
                jsr out2                ; output

mdr7            inc ls_found
                lda ls_lines            ; window filled?
```

```
                cmp #16
                beq mdr6                ; yes, finished

mdr2            lda #$20                ; no, next entry
                jsr bindex
                sta ls_index
                bcc mdr9

; ------------------------------------------------

                lda ls_found            ; any files?
                beq mdr4
mdr6            inc blocks              ; count blocks
                inc ls_cblocks

mdr4            lda dirbuf+1            ; next block
                ldy blocks
                sta blocks,y            ; store number
                tax
                lda dirbuf              ; last in chain?
                beq mdr5                ; yes, finished
                sta ls_track
                sta ftrack,y
                lda ls_lines            ; window full?
                cmp #16
mdr5            rts
;
mdr9            jmp gd_makedir

; ------------------------------------------------ Move index into Dirblock

bindex          sta sc_merk
bin2            lda ls_index
                clc
                adc sc_merk
                tax
                rts

; ------------------------------------------------ Prepare output

out1            inc sc_zl               ; next line
                inc ls_lines
                lda sc_zl               ; get line
                ldx dspalte             ; get indent
                ldy #16                 ; width
                jsr gd_setpar           ; set parameters
                jmp gd_clrline          ; clear line

; ------------------------------------------------ Convert PETSCII to Screencode

gd_cnvbc        ldy #$00
l99f8           lda sc_movetab,y
                beq cnva0
                bmi l9a19
                cmp #$20
```

```
              bcc l9a0b
              cmp #$60
              bcc l9a09
              and #$df
              bne l9a0b
l9a09         and #$3f
l9a0b         sta sc_movetab,y
              iny
              cpy sc_br
              bne l99f8
l9a13         lda #$00
              sta sc_movetab,y
              rts
;
l9a19         cmp #$a0                  ; finished if $a0
              beq l9a13
              and #$7f
cnva0         ora #$40
              bne l9a0b

; ---------------------------------------------- Pointer Management

gd_sproff     lda #0
              .by $2c
gd_spron      lda #$03
              sta $d015
              rts

; ---------------------------------------------- Read Disk

gd_onebyte    jsr basin
              ldy #0
              ldx status
              rts

; ---------------------------------------------- Open for Input

fopen         jsr gd_sproff
              jsr open
              ldx #13
              jmp chkin

; ---------------------------------------------- RAM active?

gd_testram    lda ls_drive
              cmp #12
              rts

; ---------------------------------------------- Switch REU

gd_swapd      lda #$92
              ldx #6
              .by $2c

; ---------------------------------------------- RAM Load
```

```
gd_reu      ldx #13
            sta combuf
setreu      ldy #6
sre0        lda swapw,x
            sta reubase+2,y
            dex
            dey
            bpl sre0
            lda combuf
            sta reucom
            rts

; ---------------------------------------------

magic2      .by $ad

; ---------------------------------------------
;Screenlists
; ---------------------------------------------

filelst     .by 0
            .by 0,3,34,25
            .by 1
            .by 0,0
;
dirwind     .by 1,4,18,18
            .by $60
            .by <(se_select),>(se_select)
;
            .by 19,4,18,3
            .by $60
            .by <(se_input),>(se_input)
;
            .by 11,22,3,4
            .by $d0
            .by <(se_scdwn),>(se_scdwn)
            .by $1e,$00                 ; Up Arrow
;
            .by 15,22,3,4
            .by $40
            .by <(se_scup),>(se_scup)
;
            .by 2,23,13,7
            .by $60
            .by <(se_units),>(se_units)
;
            .by 10,25,11,3
            .by $d0
            .by <(se_delete),>(se_delete)
            .ts "Delete@"
;
            .by 13,25,11,3
            .by $d0
            .by <(se_load),>(se_load)
```

```
reqtype     .ts "Load@"
;
            .by 16,25,11,3
            .by $d0
            .by <(se_ldcan),>(se_ldcan)
            .ts "Cancel@"

            .by 19,25,11,3,$a0,0,0
filtype     .ts "  *  @"

            .by 22,3,34,3,$49,<(sevch15),>(sevch15)
;
            .by $c0,16,22,1,$1f,$00        ; down Arrow
            .by $c0,0,26,5
            .ts "Units@"
            .by $c0,2,26,2
            .ts "8:@"
            .by $c0,3,26,2
            .ts "9:@"
            .by $c0,4,25,3
            .ts "10:@"
            .by $c0,5,25,3
            .ts "11:@"
            .by $c0,6,24,4,
            .ts "RAM:@"
            .by $80

; ---------------------------------------------

ldr         .tx "ldr.*"
            .by 0
svr         .tx "svr.*"
            .by 0
mod         .tx "mod.*"
            .by 0
joker       .tx "  *  "
            .by 0
nodrive     .ts "Off.@"
drvtype     .tx "1571"
ramtype     .tx "1700"
            .tx "c64 "
            .tx "vdc1"
            .ts "PFox"
            .tx "cmd "
;
messu1      .tx "u1:13 0 "
logo        cmp gd_soffx
            bcs logo2
            lda #0
logo2       rts
dirchan     .tx "#"
dspalte     .by 5
;
hirs        .ts "Hires@"
mult        .ts "Multi@"
```

```
; ------------------------------------------------

exlst           .by $00
                .by 9,7,27,5
                .by $01
                .by 0,0
                .by 10,8,8,3
                .by $c0
                .by <(se_ende),>(se_ende)
                .ts " Exit @"
                .by 10,25,8,3
                .by $c0
                .by <(se_cancel),>(se_cancel)
                .ts "Cancel@"
                .by $c0
                .by 10,16,7
                .ts "Really?@"
                .by $80

; ------------------------------------------------
magic3          .by $ad
;
; ------------------------------------------------
; screen5 (5th revision of GoDot's Screen)
; ------------------------------------------------

mainlst         .by $93
                .by $00,$00,$14,$05
                .by $01
modtype         .by $40,$80
;
                .by $00,$06,$0e,$03
                .by $c6
                .by <(ev_ldr),>(ev_ldr)
intype          .ts "    none    @"
;
                .by $02,$06,$0e,$03
                .by $ce
                .by <(ev_svr),>(ev_svr)
outtype         .ts "    none    @"
;
                .by $05,$00,$14,$09
                .by $01
                .by $00,$00
;
                .by $07,$06,$0e,$03
                .by $cf
                .by <(ev_dith),>(ev_dith)
dithtype        .ts "    Off     @"
;
                .by $09,$00,$14,$03
                .by $cb
                .by <(ev_balance),>(ev_balance)
                .ts "Balancing@"
```

```
;
                .by $0b,$00,$14,$03
                .by $cb
                .by <(ev_pal),>(ev_pal)
                .ts "Palette@"
;
                .by $12,$15,$13,$07
                .by $01
                .by $00,$00
;
                .by $14,$1a,$0e,$03
                .by $cf
                .by <(ev_mod),>(ev_mod)
efftype         .ts "    none    @"
;
                .by $16,$15,$13,$03
                .by $cb
                .by <(ev_exec),>(ev_exec)
                .ts "Execute@"
;
                .by $00,$15,$13,$07
                .by $01
                .by $00,$00
;
                .by $00,$15,$08,$03
                .by $c4
                .by <(ev_disp2),>(ev_disp2)
                .ts "Redisp@"
;
                .by 0,29,11,3
                .by $d6
                .by <(ev_info),>(ev_info)
                .ts "GoDot!@"
                .by 2,29,11,$03
                .by $ce
                .by <(ev_save),>(ev_save)
txtsave         .ts "Save@"
;
                .by 4,29,11,3,$ce,<(ev_exit),>(ev_exit)
                .ts "Exit@"
;
                .by 3,21,$08,$04
                .by $c7
                .by <(ev_load),>(ev_load)
txtload         .ts "Load@"
;
                .by $0e,$00,$14,$0b
                .by $01
                .by $00,$00
;
                .by $10,$00,$14,$03
                .by $ca
                .by <(ev_sccont),>(ev_sccont)
scrmode         .ts "Hires@"
;
```

```
                .by $12,$0e,$06,$03
                .by $ce
                .by <(ev_cols),>(ev_cols)
numcols         .tx "  2 @"
;
                .by 20,14,6,3,$ce,<(ev_area),>(ev_area)
areatype        .ts "Full@"
;
                .by $16,$00,$14,$03
                .by $cb
                .by <(ev_display),>(ev_display)
                .ts "Display@"
;
                .by $07,$15,$13,$0b
                .by $81,0,0
                .ts "Image Information@"
;
                .by 11,29,11,7,$4d,<(ev_prviu),>(ev_prviu)
;
                .by $c0,0,0,5
                .ts "Load:@"
;
                .by $c0,2,0,5
                .ts "Save:@"
;
                .by $c0,5,2,14
                .ts "Color Controls@"
;
                .by $c0,7,0,5
                .ts "Dith:@"
                .by $c0,20,21,5
                .ts "Inst@"
                .by $c0,18,22,15
                .ts "Image Operators@"
;
                .by $c0,14,1,15
                .ts "Screen Controls@"
;
                .by $c0,18,1,8
                .ts "Colors@"
;
                .by $c0,20,1,9
                .ts "Exec Area@"
                .by $c0,9,21,17
picname         .ts "Untitled       @"
                .by $c0,11,21,7
iloader         .ts "       @"
                .by $c0,13,21,7
imode           .ts "       @"
                .by $c0,15,21,7
idrive          .ts "       @"
                .by $80

; ------------------------------------------------
```

```
list            .wo $ffff
;
mloop           lda sc_screenvek
                sta list
                lda sc_screenvek+1
                sta list+1
                jsr screen
                jsr gd_eloop
                lda list
                sta sc_screenvek
                lda list+1
                sta sc_screenvek+1
                rts

; ------------------------------------------------

                .en
```