GoDot

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

```
.petscii
; 2.4.93
; 1.11.00, complete text revision

                .ob "god_upmem,p,w"
;       .li 1,4,7
;

                .eq pport               =$01            ; usually $36
                .eq sc_vekt20           =$20
                .eq sc_zl               =$22
                .eq sc_sp               =$23
                .eq sc_br               =$24

                .eq sc_loop             =$a6
                .eq ls_vekta8           =$a8
                .eq gr_nr               =$aa
                .eq sc_screentab        =$ab
                .eq sc_texttab          =$ae
                .eq sc_merk             =$b0
                .eq dest1               =$b2

                .eq sc_taste            =$c6
                .eq gr_basepal          =$02b0
                .eq gd_tbuffer          =$0277
                .eq sc_shadow           =$0334
                .eq sc_chgcol           =$0338
                .eq gr_defpals          =$03c0
                .eq defphi              =$03ce

                .eq sc_screenvek        =$0f88
                .eq sc_maincolor        =$0f8e
                .eq gr_cmode            =$0f95
                .eq gr_howmany          =$0f96
                .eq gr_picked           =$0f97
                .eq gr_palette          =$0f98
                .eq gr_redisp           =$0fbd
                .eq gr_bkcol            =$0fc6

                .eq gd_initmove         =$1090
                .eq gd_fcol             =$110c
                .eq fi0                 =$1114
                .eq gd_invert           =$1125
                .eq gd_setpos           =$11cc
                .eq gd_screen           =$11e0
                .eq gd_trim             =$1324
                .eq gd_endirq           =$13e5
                .eq gd_version          =$1415
                .eq gd_eloop            =$1423
```

```
                .eq savescvek           =$143f
                .eq dithtype            =$1e30

                .eq palundo             =$be00
                .eq palcanc             =$be10
                .eq paldflt             =$be20

                .eq grafreg             =$d011
                .eq multreg             =$d016
                .eq charreg             =$d018
                .eq border              =$d020
                .eq backgr              =$d021

                .eq cstart              =$fe0b
                .eq vram1               =$de0b
                .eq vram2               =$ee0b
```

```
                .ba $c000

; ------------------------------------------------ Header

                jmp start
;
                .by $20,$00,$00
                .wo modend
                .wo 0
                .tx "GoDot f000 slot4"
created         .tx "1.14"
                .tx "16.08.93"          ; date changed to 18.02.96
                                        ; (start of NA distribution)
                .tx "A.Dettke/W.Kling"
;
; ------------------------------------------------ Module Management

start           stx pdis                ; .x=0; clear flag (request from palette requester)
                ldx sc_screenvek        ; save vector to main screen screenlist
                stx list
                ldx sc_screenvek+1
                stx list+1
                ldx gr_nr               ; get function number
                beq jevbal              ; 0: Balancing
                dex
                beq jevpal              ; 1: Palette
                dex
                beq jevdispl            ; 2: Display
                dex
                beq jevdith             ; 3: Dithermode
                dex
                beq jevarea             ; 4: Toggle Clip/Full
                dex
                beq jevinfo             ; 5: GoDot Guru
                dex
                beq jevprviu            ; 6: Switch Preview on
```

```
            dex
            beq jevdisp2            ; 7: Redisplay
            clc
            rts

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

jevdispl    jmp ev_display
jevpal      jmp ev_pal
jevbal      jmp ev_bal
jevdith     jmp ev_dith
jevarea     jmp ev_area
jevprviu    jmp ev_prviu
jevdisp2    jmp ev_disp2
jevinfo     jmp ev_info

; ------------------------------------------------ Show graphics

redisplay   ldx gr_palette          ; bordercolor is palette entry 0
            stx border
            jsr setcolors           ; reget colors
            lda gr_cmode            ; which graphics mode?
            beq dp4
            lda #$18                ; multi when 2
            sta multreg
dp4         lda #$1b                ; hires when 0
            sta charreg
            lda #$3b
            sta grafreg
            lda gr_redisp           ; image already rendered?
            bne dp3
            sta sc_taste            ; yes, clear keypressed
            pla                     ; clear stack
            pla
dp1         lda sc_taste            ; wait for key or STOP
            ora stop
            beq dp1
            jsr getcolors           ; save colors
            jsr tmode               ; switch textmode on
            lda sc_screenvek        ; save vector to current screenlist
            pha
            lda sc_screenvek+1
            pha
            lda pdis                ; is palette requester active?
            beq dp5
            ldx list                ; yes, rebuild main screen
            ldy list+1
            jsr gd_screen
dp5         pla                     ; rebuild palette requester
            tay
            pla
            tax
            jsr gd_screen
dp3         clc                     ; finished (stay in list)
            rts
```

```
; ------------------------------------------------ Switch Textmode on

tmode       ldx #$13
            lda #$1b
            stx charreg
            sta grafreg
            lda #$08
            sta multreg
            lda sc_maincolor
            sta border
            sta backgr
            rts

; ------------------------------------------------ Graphics Colors Management

setcveks    sei                     ; called from GETCOLORS
            lda #$35                ; set C64 to IO off
            sta pport
            lda #>(cstart)          ; first: save 1000 color RAM nibbles to loader area
            ldx #$d8                ; from $d800
            bne scv0
setbveks    lda #>(vram1)           ; second: save upper 500 video RAM bytes to saver
area
            ldx #$04                ; from $0400
            dec pport               ; set C64 completely to RAM
scv0        stx dest1+1
            ldy #$00
            sty sc_merk
            sty dest1
            dey
setlast     sty gr_bkcol            ; third: save lower 500 video RAM by. to mod area
            ldy #<(cstart)
            sty sc_vekt20           ; vector to data
            sta sc_vekt20+1
            lda #<(500)             ; counter
            sta ls_vekta8
            lda #>(500)
            sta ls_vekta8+1
            ldy #$00
            rts

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

count       inc sc_vekt20           ; increment source data vector
            bne cou5
            inc sc_vekt20+1
cou5        inc dest1               ; increment destination data vector
            bne cou6
            inc dest1+1
cou6        lda ls_vekta8           ; decrement counter
            bne cou7
            dec ls_vekta8+1
cou7        dec ls_vekta8
            lda ls_vekta8           ; until zero
```

```
                ora ls_vekta8+1
                rts

; ---------------------------------------------- Save Graphics Colors

getcolors       jsr setcveks            ; set vectors accordingly
stco            lda (dest1),y           ; compress $d800-nibbles
                sta sc_merk
                inc dest1
                bne stc0
                inc dest1+1
stc0            lda (dest1),y
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                lda sc_merk
                sta (sc_vekt20),y       ; and buffer them
                jsr count
                bne stco

                jsr setbveks            ; buffer video RAM, part 1
stc1            lda (dest1),y
                sta (sc_vekt20),y
                jsr count
                bne stc1
                ldy gr_bkcol            ; flag: last time?
                bpl scv1                ; yes, finished

                ldy #$00                ; buffer video RAM, part 2
                lda #>(vram2)
                jsr setlast
                beq stc1                ; unconditional branch

scv1            lda #$36                ; set C64 to BASIC off
                sta pport
                cli
                lda backgr              ; set graphics background color value
                and #$0f
                sta gr_bkcol
                rts

; ---------------------------------------------- Restore Graphics Colors

setcolors       lda gr_bkcol            ; restore background color
                sta backgr
                pha                     ; save for last time

                jsr setcveks            ; set vectors accordingly
stc2            sty sc_merk             ; decompress color RAM nibbles
                lda (sc_vekt20),y
```

```
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                lsr
                rol sc_merk
                sta (dest1),y           ; and store them
                inc dest1
                bne stc3
                inc dest1+1
stc3            lda sc_merk
                sta (dest1),y
                jsr count
                bne stc2

                jsr setbveks            ; restore video RAM, part 1
stc4            lda (sc_vekt20),y
                sta (dest1),y
                jsr count
                bne stc4

                ldy gr_bkcol            ; flag: last time?
                bpl scv1                ; yes, finished

                pla                     ; restore video RAM, part 2
                tay
                lda #>(vram2)
                jsr setlast
                beq stc4

; ------------------------------------------------ Palette Management

tabigad         .by <(pickbox),>(pickbox)               ; text output routines
                .by <(palbox),>(palbox)
                .by <(pick),>(pick)
;
settab          lda #<(tabigad)
                sta sc_texttab
                lda #>(tabigad)
                sta sc_texttab+1
                rts
;
gettab          lda (sc_texttab),y
                sta sc_screentab
                iny
                lda (sc_texttab),y
                sta sc_screentab+1
                iny
                sty sc_merk
                stx sc_merk+1
                lda #$00
                tax
                tay
```

```
gt0         jsr gd_setpos
            cpx #$04
            bne gt0
            jsr gd_trim
            jmp gd_initmove

; ------------------------------------------------ Subevent: Display from Palette Requester

se_paldis   jsr ev_display          ; switch graphics mode on

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

palcols     inx                     ; .x was 0; set index to Normal Text color
            inx
            jsr gd_fcol             ; colorize "Pick Color." message

            ldy #$00                ; care for textmode
            sty sc_loop
            jsr settab              ; set output vector

            ldx #$02                ; prepare palette gadgets
ep0         jsr gettab
            jsr gd_invert           ; invert them (to show colors later)
            ldy sc_merk
            ldx sc_merk+1
            dex
            bne ep0

ppal        jsr settab              ; re-set output vector
            ldy #$00                ; start with upper palette gadget
            sty sc_chgcol           ; and with black
            ldx #$0f                ; 16 colors
            jsr gettab
            ldy #$02                ; width: 2 tiles
            sty sc_br
ep1         ldx #$04                ; index to color
            jsr gd_fcol             ; colorize
            inc sc_sp               ; increment tile offset
            inc sc_sp
            inc sc_chgcol           ; increment color (standard palette)
            dec sc_merk+1           ; count colors
            bpl ep1

            ldy #$02                ; now lower palette gadget
            jsr gettab
            ldx #$00                ; index into GoDot palette
            ldy #$02                ; width again: 2 tiles
            sty sc_br
ep2         lda gr_palette,x        ; get color entry
            sta sc_chgcol
            ldx #$04                ; index to color
            jsr gd_fcol             ; colorize
            inc sc_sp               ; increment tile offset
            inc sc_sp
            inc sc_merk+1           ; increment color index (GoDot palette)
```

```
                ldx sc_merk+1           ; last valid entry?
                cpx gr_howmany
                bcc ep2                 ; no, loop

ep3             ldx gr_picked           ; color selected?
                bmi ep44                ; no, finished

                stx sc_chgcol           ; yes, colorize "Picked" gadget
ep9             lda #$00                ; care for textmode
                sta sc_loop
                jmp eps7                ; continue there
;
ep44            clc
                rts

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

ev_pal          jsr savescvek           ; save vector to main screenlist
                sta pdis                ; set Flag: rebuild Palette requester

                lda gr_howmany          ; compute width of GoDot palette gadget
                asl
                tax
                inx
                inx
                stx palbox+2

                ldx #<(pallst)          ; show requester
                ldy #>(pallst)
                jsr gd_screen

                ror                     ; .a=$80
                sta gr_picked           ; set “Picked invalid”
                jsr palcols             ; show palettes

                ldx #$0f                ; initialize palette buffers
ep5             lda gr_palette,x        ; get 16 current palette values
                sta palundo,x           ; store to undo area
                sta palcanc,x           ; store to nothing changed area
                lda gr_basepal,x        ; get base palette values
                sta paldflt,x           ; store to default palette area
                dex
                bpl ep5
                jmp gd_eloop            ; wait for further events

; ------------------------------------------------ Subevent: Cancel Palette Requester

se_pcanc        lda #$00                ; flag: don't re-render
                sta gr_redisp

                ldx #$1f                ; get PALCANC
ep7             ldy #$0f
ep6             lda palundo,x
                sta gr_palette,y        ; and restore palette
                dex
```

```
            dey
            bpl ep6

            jsr ep3                 ; care for Picked gadget
ep61        lda #$80                ; set Picked value to invalid
            ora gr_picked
            sta gr_picked

; ---------------------------------------------- Subevent: Accept Palette Changes

se_pacc     sec                     ; leave requester to main screen
            rts

; ---------------------------------------------- Subevent: Set Default Palette

se_pdeft    ldx gr_howmany          ; get number of colors to be rendered
            cpx #$10                ; is 16?
            bne dpl1
            lda #$ff                ; yes, set flagword to $ffff
            sta sc_merk+1
            bne dpl2
dpl1        dex                     ; no, index minus 2
            dex
            lda defphi,x            ; get flagword for this particular palette
            sta sc_merk+1
            lda gr_defpals,x
dpl2        sta sc_merk

            ldy #$00                ; palettes are coded bitwise
            ldx #$00                ; index into base palette
dp          asl sc_merk+1           ; shift flags left
            rol sc_merk
            bcc dps
            lda gr_basepal,x        ; bit set: get associated color
            sta paldflt,y           ; enlist in current palette
            iny
            cpy gr_howmany          ; until all colors
            beq dpok

dps         inx                     ; increment index
            cpx #$10                ; until 16
            bne dpl

dpok        ldx #$2f                ; move palettes
ep8         jsr ep7
            jsr eps8                ; show new (default) palette
ep4         clc                     ; finished
            rts

; ---------------------------------------------- Subevent: Undo Change to Palette

se_pundo    ldx #$0f                ; restore current palette
            bne ep8                 ; and display

; ---------------------------------------------- Subevent: Pick a Color
```

```
se_ppick    ldx gr_picked           ; first get current picked color
            lda sc_merk             ; get screen position: row
            sta sc_zl
            lda sc_merk+1           ; and column
            sta sc_sp
            inc sc_loop             ; care for colormode
            jsr gd_initmove         ; compute address of this position in color RAM
            lda (sc_vekt20),y       ; .y=0; retrieve picked color
            and #$0f
            sta gr_picked           ; store value
            sta sc_chgcol
            sta sc_merk+1
            jsr settab              ; get vector address

            txa                     ; get current picked color
            bpl ep91                ; repeatedly picked a color?

            dec sc_loop             ; no, care for textmode
eps7        ldy #$04                ; index to gadget descriptor address
            jsr gettab              ; compute destination
            jsr gd_invert           ; invert gadget
            ldx #$04                ; index to new color (points to sc_chgcol)
            jsr gd_fcol             ; and colorize it
            beq ep4                 ; unconditional branch, finished

ep91        ldx sc_merk+1           ; yes, get picked color
            ldy #$04                ; compute destination
            jsr gettab
            lda sc_merk+1           ; get color
            jsr fi0                 ; show color
            beq ep4                 ; unconditional branch; finished

; ---------------------------------------------- Subevent: Change GoDot Palette

se_pset     lda gr_picked           ; get new color value
            bmi ep4                 ; invalid, finished

            jsr setundo             ; save current palette
            sec
            lda sc_merk+1           ; get current mousepointer column
            and #$fe                ; make even
            sta sc_merk+1
            sbc sc_sp               ; subtract gadget position (column)
            lsr                     ; divide by 2
            tax                     ; take as an index into palette

            lda sc_merk+1           ; current column pointed to (but made even)
            sta sc_sp
            lda #$02                ; width: 2
            sta sc_br
            lda gr_picked           ; retrieve color
            sta gr_palette,x        ; enlist in current palette
            jsr ep61                ; set picked color to invalid
            jsr eps7                ; show color in Picked gadget
```

```
eps8        dex                     ; .x=$ff now
            stx gr_redisp           ; flag: re-render graphics
            jmp ppal                ; show new palette

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

setundo     ldx #$0f                ; save current palette
su1         lda gr_palette,x
            sta palundo,x
            dex
            bpl su1
            rts

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

            .eq sc_ho               =$25
            .eq src                 =$39
            .eq col00               =$3b
            .eq nr0                 =$40
            .eq max                 =$44
            .eq rstart              =max
            .eq rend                =max+1
            .eq cols                =$46
            .eq zp1                 =$47
            .eq xbuf                =$48
            .eq rstep               =xbuf
            .eq hmzp                =$49
            .eq cmzp                =$4a
            .eq pbuf                =$4b

            .eq sc_pos              =$a4
            .eq source              =$a6
            .eq dest                =$a8
            .eq vblock              =$aa
            .eq bitcnt              =$ac
            .eq source1             =$ae
            .eq hold                =$b4
            .eq vblock1             =$b5

            .eq sc_lastclipzl       =$033c
            .eq gr_orderedpat       =$0340
            .eq gr_btab             =$03b0
            .eq gr_qtab             =$0ee0
            .eq gr_pattern          =$0f00
            .eq gr_dither           =$0fbc
            .eq sc_clipzl           =$0fbe
            .eq sc_clipsp           =$0fbf
            .eq sc_clipbr           =$0fc0
            .eq sc_clipho           =$0fc1
            .eq sc_clipped          =$0fc2
            .eq sc_stop             =$0fc5

            .eq gd_xtxout1          =$1461
            .eq areatype            =$1ef4
```

```
                .eq hist1               =$3f40
                .eq hist                =$3f50
                .eq etab                =$3f70
                .eq htab                =$3f80
                .eq htab0               =$3f90
                .eq rtab                =$3fb0

                .eq oszi3               =$d40e
                .eq wave                =$d412
                .eq filter              =$d418
                .eq rausch              =$d41b

; ------------------------------------------------ Event Re-Display Graphics

ev_disp2        lda gr_redisp           ; save current re-display status
                pha
                ldx #$01                ; force re-display
                stx gr_redisp
                dex                     ; clear STOP key
                stx sc_stop
                jsr redisplay           ; re-display
                jsr dp1                 ; wait for keypress
                pla                     ; restore redisp status
                sta gr_redisp
                rts

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

ev_display      jsr redisplay           ; display graphics

                lda #$80                ; initialize noise (random) values
                sta filter
                sta oszi3
                sta oszi3+1
                lda #$00
                sta wave
                sta sc_stop
                lda #$81
                sta wave

                lda gr_howmany          ; number of colors to be rendered
                sta hmzp                ; store in zeropage
                lda gr_cmode            ; get graphics mode
                sta cmzp                ; and store to zeropage
                jsr setvecs             ; initialize all vectors and values
                jsr setclip             ; compute startaddress of clip

                lda dest                ; destination ($2000)
                ldx dest+1
                sta dest1
                stx dest1+1
                jsr clrclip             ; clear clip area and provide histograms

                lda source              ; source ($4000)
                ldx source+1
```

```
                sta source1
                stx source1+1
                lda vblock              ; video RAM ($0400)
                ldx vblock+1
                sta vblock1
                stx vblock1+1

                ldx gr_dither           ; set vector of current dither routine
                lda routtablo,x
                sta sc_pos
                lda routtabhi,x
                sta sc_pos+1

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

mainpix         lda sc_clipho           ; height
                sta sc_ho
dhb4            lda sc_clipbr           ; width
                sta sc_br

dhb1            clc
                lda #$00                ; init indeces
                sta sc_merk
                sta sc_merk+1
dhb01           jsr histo               ; compute histogram for 1 tile

dhb0            ldy sc_merk             ; get first byte (two 4bit pixels)
                lda (source),y
                pha
                ldy cmzp                ; color mode
                bne mcskip              ; multi?
                lsr                     ; no hires, get first 4b pixel (left)
                lsr
                lsr
                lsr
mcskip          and #$0f                ; isolate 4b pixel
                tax                     ; care for balancing (gray: contrast/brightness)
                lda gr_btab,x
                ldy sc_merk+1           ; index for "Ordered" pattern
                jsr pixel               ; dither, returns a bitpattern
                and bitcnt              ; isolate bitmap pixel position
                ora (dest),y            ; and set bm pixel
                sta (dest),y
                lsr bitcnt              ; next bm pixel position

                pla                     ; re-get 4bit byte (right pixel)
                and #$0f                ; isolate 4b pixel
                tax
                lda gr_btab,x           ; balancing, .y still valid
                jsr pixel               ; dither, returns a bitpattern
                and bitcnt              ; isolate bm pixel
                ora (dest),y            ; and set
                sta (dest),y

                inc sc_merk             ; inc index into tile
```

```
                lsr bitcnt              ; next pixel position
                bcc dhb0

                ror bitcnt              ; 8 pixels finished, reset mask
                inc sc_merk+1           ; inc index into Ordered pattern
                lda sc_merk+1
                cmp #$08                ; 8 scanlines finished?
                bcc dhb01

                inc vblock              ; yes, next tile:
                bne dhb21
                inc vblock+1

dhb21           clc                     ; add 8 to dest vector
                adc dest
                sta dest
                bcc dhb2
                inc dest+1
dhb2            clc                     ; add 32 to source vector
                lda #$20
                adc source
                sta source
                bcc dhb3
                inc source+1

dhb3            lda sc_stop             ; STOP pressed?
                bne break               ; yes, break

                dec sc_br               ; decrement width value
                bne dhb1                ; until zero

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

                clc                     ; to tile below first one:
                lda vblock1             ; add 40 to video RAM
                adc #$28
                sta vblock1
                sta vblock
                lda vblock1+1
                adc #$00
                sta vblock1+1
                sta vblock+1
                clc                     ; add 320 to dest
                lda dest1
                adc #$40
                sta dest1
                sta dest
                lda dest1+1
                adc #$01
                sta dest1+1
                sta dest+1
                clc                     ; add 1280 to source
                lda source1+1
                adc #$05
                sta source1+1
```

```
                sta source+1
                lda source1
                sta source

                dec sc_ho               ; dec height
                beq dhb9                ; until zero

                jmp dhb4                ; else loop

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

break           sta gr_redisp           ; force re-render
                sta sc_taste            ; simulate keypressed
dhb9            jmp dp1                 ; end sequence (graphics off...)

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

pixel           jmp (sc_pos)            ; entry for dither routines

; ------------------------------------------------ Compute Clip data

setclip         lda sc_clipzl           ; row of clip
                beq scp1                ; zero?

                sta sc_zl               ; no, increment video RAM by 40
scp4            clc
                lda vblock
                adc #$28
                sta vblock
                bcc scp2
                inc vblock+1
scp2            clc                     ; increment dest by 320
                lda dest
                adc #$40
                sta dest
                lda dest+1
                adc #$01
                sta dest+1
                clc                     ; increment source by 1280
                lda source+1
                adc #$05
                sta source+1
                dec sc_zl               ; as many times as vertically indented
                bne scp4

scp1            lda sc_clipsp           ; column of clip
                beq scp5                ; zero?

                sta sc_sp
scp8            inc vblock              ; no, inc video by 1
                bne scp81
                inc vblock+1
scp81           clc                     ; inc dest by 8
                lda dest
                adc #$08
```

```
            sta dest
            bcc scp6
            inc dest+1
scp6        clc                     ; inc source by 32
            lda source
            adc #$20
            sta source
            bcc scp7
            inc source+1
scp7        dec sc_sp               ; as many times as horizontally indented
            bne scp8

scp5        rts

; ------------------------------------------------ Dith type: No Dither

dithoff     tax                     ; regard quantization table
            lda gr_qtab,x
            tax                     ; get appropriate pixel pattern ($0 or $f)
            lda htab,x
            rts

; ------------------------------------------------ Dith type: Ordered

dithhab     sta zp1
            tax                     ; regard quantized colors
            lda rtab,x
            ora offs,y              ; compute offset into pattern
            tax
            lda gr_orderedpat,x     ; get pixel pattern
            jmp dh1                 ; regard graphics mode

; ------------------------------------------------ Dith type: Pattern

dithpat     sta zp1
            sty hold                ; save .y (pattern index)
            tax
            lda rtab,x              ; regard quantized colors
            asl                     ; times 8
            asl
            asl
            ora hold                ; regard index
            tax
            lda gr_pattern,x        ; get pixel pattern

dh1         ldx cmzp                ; grahics mode?
            beq dp0
            and #$0f                ; if multi:
            tax                     ; get appropriate pixel pattern
            lda mcpat,x
dp0         ldx zp1                 ; get original value
            and bitcnt              ; isolate bm pixel
            beq bgr                 ; not set?
            bne fgr                 ; set?
```

```
; ------------------------------------------------ Dith type: Random

dithrnd     tax
            lda rtab,x          ; regard quantized colors
            beq bgr
            cmp #8              ; threshold: 8
            bcs fgr

            lda rtab+1,x        ; regard neighboring color
            beq dr0
            cmp #8
            bcc bgr

dr0         jsr getrnd          ; get random value
rold        lda zp1
            and #1
            beq bgr             ; even?
            bne fgr             ; odd?

; ------------------------------------------------ Dith type: Noise

dithnoi     tax
            lda rtab,x          ; regard quantized colors
            beq bgr
            sta hold            ; no threshold
            jsr getrnd          ; get random value
nold        lda hold
            cmp zp1             ; bigger?
            bcs fgr

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

bgr         lda htab0,x         ; pixel not set, get pixel pattern
            rts
;
fgr         lda htab,x          ; pixel set, get pixel pattern
            rts

; ------------------------------------------------ Get Random Value

getrnd      lda cmzp            ; graphics mode?
            beq gr00
            lda bitcnt          ; multi, isolate right pixels
            and #$55
            bne gr01
gr00        lda rausch          ; hires, get random value
            and #15             ; delimit to 15
            sta zp1
gr01        rts

; ------------------------------------------------ Get start values

setvecs     lda #$00
            sta sc_taste        ; clear keyboard
            sta gr_redisp       ; clear re-render flag
```

```
                sta vblock              ; low bytes all $00
                sta source
                sta dest
                lda sc_clipped          ; any clip set?
                bne sv0                 ; yes, don't change values

                sta sc_clipzl           ; no, values to 0,0,40,25
                sta sc_clipsp
                lda #$19
                sta sc_clipho
                lda #$28
                sta sc_clipbr

sv0             lda #$80                ; init bitmask for pixel pattern
                sta bitcnt
                lsr                     ; source from $4000
                sta source+1
                lsr                     ; dest from $2000
                sta dest+1
                lda #$04                ; video from $0400
                sta vblock+1
                rts

; ------------------------------------------------ Clear Clip Area

clrclip         ldx sc_clipho           ; height of clip
                ldy #$00

cc2             sty sc_pos+1
                lda sc_clipbr           ; width of clip times 8
                asl
                asl
                asl
                bcc cc3
                inc sc_pos+1
cc3             sta sc_pos

                lda dest                ; save dest
                pha
                lda dest+1
                pha

cc0             tya                     ; .y=0
                sta (dest),y            ; clear bytes (to black)
                inc dest
                bne cc1
                inc dest+1
cc1             lda sc_pos              ; count width
                bne cc4
                dec sc_pos+1
cc4             dec sc_pos
                lda sc_pos              ; until 0
                ora sc_pos+1
                bne cc0
```

```
            pla
            sta dest+1
            pla
            clc                 ; add 320 to dest
            adc #$40
            sta dest
            lda dest+1
            adc #$01
            sta dest+1
            dex                 ; decrement height
            bne cc2

            lda dest1           ; restore dest
            ldx dest1+1
            sta dest
            stx dest+1

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

            lda gr_dither       ; any dither?
            beq hist0           ; no, compute histograms

            lda hmzp            ; yes, get number of colors/grays
            cmp #$02            ; b&w?
            bne makeetab        ; no, create etab and compute histograms

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

            lda cmzp            ; yes, b&w, graphics mode?
            beq mcb
            lda #$55            ; multi, get pattern $55
            .by $2c
mcb         lda #$ff            ; hires, pattern $ff
            ldy #$0f

hm2l        sta htab,y          ; put to table of pixel patterns
            eor #$ff
            sta htab0,y
            eor #$ff
            tax
            tya
            sta rtab,y          ; put to table of quantized colors
            txa
            dey
            bpl hm2l

            lda gr_palette+1    ; get neighboring color
            asl                 ; shift to upper nibble
            asl
            asl
            asl
            ora gr_palette      ; OR with color
            sta pbuf            ; store value (faster rendering now)
            rts
```

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

makeetab    ldy #$01
            ldx #$00
etl         lda gr_qtab,y           ; compare neighboring values in quantization table
            cmp gr_qtab-1,y
            beq ets                 ; if equal, ignore
            tya
            sta etab,x              ; if not, set position of change to etab
            inx
ets         iny
            cpy #$10
            bne etl

; ------------------------------------------------ Make Histograms

hist0       ldx #$2f                ; clear histograms
            stx col00+4
            jsr clh1

            sta src                 ; start from $4000
            lda #$40
            sta src+1
            lda cmzp                ; get graphics mode
            beq hi01
            lda #$aa                ; is multi: change bits
hi01        sta bits+2
            beq hready              ; finished when hires

            lda sc_clipped          ; any clip?
            beq hloop               ; no, proceed

            ldy hmzp                ; yes, number of colors/grays
            dey
cll1        lda gr_palette,y        ; background color part of palette?
            cmp backgr
            beq cls1                ; yes, store index
            dey
            bpl cll1
            ldy #$20                ; no, set flag
cls1        sty bktmp
hready      rts                     : no histograms when clipped

; ------------------------------------------------ Histograms for Multi Mode

hloop       jsr clrhist1            ; clear histogram1

            ldy #$1f                ; retrieve colors
l1          lda (src),y
            and #$0f
            tax
            lda gr_btab,x           ; regard balancing and quantization
            tax
            lda gr_qtab,x
            tax
```

```
            inc hist1,x
            dey
            bpl l1

            ldy #$00                ; how many colors per tile?
            ldx hmzp
            dex
            sty col00
l2          lda hist1,x
            beq s11
            iny
s11         dex
            bpl l2
            cpy #$04                ; less than 4?
            bcc s2                  ; yes, proceed

            ldx hmzp                ; no, count colors in these particular tiles
            dex
l3          lda hist1,x
            beq s3
            clc
            adc hist,x
            sta hist,x
            bcc s3
            inc hist+16,x
s3          dex
            bpl l3

s2          lda src                 ; next tile
            clc
            adc #$20
            sta src
            bcc hloop
            inc src+1
s4        lda src+1
            cmp #$bd                ; end of 4bit?
            bne hloop

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

            lda #$00                ; yes, retrieve most frequent color
            sta max
            sta max+1
            ldx hmzp
            dex
l4          lda hist+16,x
            tay
            cmp max+1
            bcc s5
            beq s51
            lda hist,x
            bcs s52
s51         lda hist,x
            cmp max
            bcc s5
```

```
            beq s5
s52         sta max
            sty max+1
            stx col00               ; contains most frequent color
s5          dex
            bpl l4

            lda col00               ; result becomes background color
            tax
            lda gr_palette,x
            stx bktmp
            sta backgr
            rts

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

clrhist1    ldx hmzp
            dex
clh1        lda #$00
l5          sta hist1,x
            dex
            bpl l5
            rts

; ------------------------------------------------ Decide which colors per tile

histo       lda gr_dither           ; any dither?
            beq dohist              ; no, histogram

            lda hmzp                ; b&w?
            cmp #2
            bne dohist              ; no, histogram

            lda pbuf                ; yes, get precomputed colors
            ldy #0
            sta (vblock),y
            rts

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

dohist      jsr clrhist1            ; clear histogram
;
            ldy #$20                ; preset values (impossible value 32)
            sty col00+3
            sty col00+2
            sty col00+1
            dey

l6          lda (source),y          ; count colors
            pha
            and #$0f
            tax
            lda gr_btab,x
            tax
            lda gr_qtab,x
```

```
                tax
                inc hist1,x
                pla
                ldx cmzp
                bne s6
                lsr
                lsr
                lsr
                lsr
                tax
                lda gr_btab,x
                tax
                lda gr_qtab,x
                tax
                inc hist1,x
s6              dey                     ; within 64 pixels
                bpl l6

                ldy cmzp                ; graphics mode
                beq s7

                ldx bktmp               ; is multi:
                stx col00
                lda #$00                ; don't regard background color
                sta hist1,x
                sta nr0
s7              ldx #$03
l7              ldy #$00
                sty max
                lda bits,x
                sta nr0,x
l8              lda hist1,y
                beq s8
                cmp max
                bcc s8
                sta max
                sty col00,x             ; get 3 most frequent colors
s8              iny
                cpy hmzp
                bne l8
                ldy col00,x             ; don't regard processed colors
                lda #$00
                sta hist1,y
                dex
                bne l7

; ---------------------------------------------- Set Colors

                ldy #$00
                lda cmzp                ; graphics mode
                beq mc1

                lda col00+1             ; is multi: get video RAM colors
                and #$0f
                tax
```

```
                lda gr_palette,x
                asl
                asl
                asl
                asl
                sta cols
                lda col00+2
                and #$0f
                tax
                lda gr_palette,x
                ora cols
                sta (vblock),y          ; and set

                lda vblock+1            ; color 3 to color RAM
                pha
                clc
                adc #$d4
                sta vblock+1
                lda col00+3
                and #$0f
                tax
                lda gr_palette,x
                sta (vblock),y
                pla
                sta vblock+1
                bne mc2                 ; unconditional branch

mc1             ldx col00+2             ; is hires: get video RAM colors
                cpx #$20                ; only 1 color in tile?
                bne ms1
                ldx col00+3             ; yes, get neighboring color for background
                inx
                cpx hmzp
                bne ms2
                dex
                dex
ms2             stx col00+2

ms1             lda col00+3             ; sort colors (when hires)
                cmp col00+2
                bpl nosort
                ldx col00+2
                sta col00+2
                stx col00+3
                txa
nosort          and #$0f
                tax
                lda gr_palette,x
                asl
                asl
                asl
                asl
                sta cols
                lda col00+2
                and #$0f
```

```
                tax
                lda gr_palette,x
                ora cols
                sta (vblock),y          ; and set
                jmp makehtab            ; make pattern table
;
mc2             ldx #$00                ; sort colors (when multi)
lx              stx xbuf
                ldy #$03
ly              lda col00,x
                cmp col00,y
                bcc s9
                beq s9
                pha
                lda col00,y
                sta col00,x
                pla
                sta col00,y
                lda nr0,x
                pha
                lda nr0,y
                sta nr0,x
                pla
                sta nr0,y
s9              dey
                cpy xbuf
                bne ly
                inx
                cpx #$03
                bne lx

; ------------------------------------------------ Create Pixelpattern Table

makehtab        lda gr_dither           ; any dither?
                bne makertab            ; yes, branch

                ldy #$ff
                sty max
                lda cmzp                ; graphics mode
                bne l9
                ldy #$01                ; is hires (.y=2)
l9              iny                     ; is multi (.y=0)
                lda col00,y             ; compute average of 2 neighboring colors
                clc
                adc col00+1,y
                lsr
                cmp #$10
                bcc s10
                lda #$0f
s10             sta max+1
                lda nr0,y
                ldx max
l10             inx
                sta htab,x              ; fill table
                cpx max+1
```

```
              bcc l10
              cpx #$0f
              bcs l12
              stx max
              cpy #$03
              bne l9
l12           rts

; ------------------------------------------------- Create transition patterns

makertab    ldx #$00
              lda cmzp                  ; graphics mode
              bne mkt0
              ldx #$02                  ; is hires (.x=2)
mkt0          ldy #$00                  ; is multi (.x=0)
mkt1          lda gr_qtab,y
              cmp col00+1,x
              bcs mkt2
              lda nr0+1,x
              sta htab,y
              lda nr0,x
              sta htab0,y
              iny
              cpy #$10
              bne mkt1
              beq mrt0
mkt2          inx
              cpx #$04
              bne mkt1
;
mrt0          ldy #$0f
              lda #$00
irl           sta rtab,y                ; set pattern
              dey                       ; (transition between two neighboring colors)
              bpl irl
;
              ldx #$00
              lda cmzp
              bne rtl0
              ldx #2
rtl0          lda col00,x
              cmp #$20
              bcs rts0
              tay
              lda etab,y
              sta rstart
              lda col00+1,x
              cmp #$20
              bcs rts0
              tay
              dey
              lda etab,y
              sta rend
              sec
              sbc rstart
```

```
                beq rts1
                tay
                lda rdtab,y
                sta rstep
                ldy rstart
                lda #$00
rtl1            clc
                adc rstep
                pha
                lsr
                lsr
                lsr
                lsr
                sta rtab,y
                iny
                cpy rend
                pla
                bcc rtl1
rts1            inx
                cpx #$04
                bne rtl0
rts0            rts

; ------------------------------------------------ Pixel Patterns

bits            .by $00,$55
                .by $aa,$ff
mcpat           .by $00,$03,$0c,$0f,$30,$33,$3c,$3f
                .by $c0,$c3,$cc,$cf,$f0,$f3,$fc,$ff
rdtab           .by $f0,$78,$50,$3c,$30,$28,$22
                .by $1e,$1b,$18,$16,$14,$12,$11,$10

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

ev_area         lda sc_clipped          ; is clipped?
                beq area0               ; no, toggle to Clip
                ldy #$07                ; yes, toggle to Full
                .by $2c
area0           ldy #$03
                ldx #$04
area1           lda atype,y
                sta areatype,x
                dey
                dex
                bne area1
                tya
                bpl area2
                ldy #$03
area3           lda sc_lastclipzl,y     ; set appropriate values
                sta sc_clipzl,y
                dey
                bpl area3
                inx
area2           stx sc_clipped          ; flag: clipped
                ldy #$06
```

```
                sty gr_redisp           ; force re-render
                jmp gd_xtxout1          ; show new status

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

                .eq ls_temp             =$a7
                .eq brttmp              =$a9
                .eq crttmp              =$aa

                .eq gr_crt              =$02a7
                .eq gr_brt              =$02a8

                .eq irq                 =$0314
                .eq sc_chgcol           =$0338

                .eq scr1                =$0432
                .eq scr2                =$0452

                .eq sprptr              =$07fa

                .eq sc_movetab          =$0fd8

                .eq gd_plus40           =$10e7
                .eq gd_backu            =$10f5
                .eq fi1                 =$111b
                .eq fi2                 =$1121
                .eq makeline            =$11ae
                .eq delay2              =$131b
                .eq gd_xtxtggl          =$146d
                .eq gd_xcnvdez          =$1473

                .eq spr2                =$3e80
                .eq spr5                =$3f40
                .eq bttab               =$3f40
                .eq btabold             =$3f50

                .eq dbuf                =$bd00
                .eq oldirq              =$be00

                .eq sprxy               =$d004
                .eq xmsb                =$d010
                .eq raster              =$d012
                .eq sprreg              =$d015
                .eq yexp                =$d017
                .eq reqirq              =$d019
                .eq rirqen              =$d01a
                .eq sprpri              =$d01b
                .eq sprmulti            =$d01c
                .eq xexp                =$d01d
                .eq sprcol2             =$d025
                .eq sprcol3             =$d026
                .eq sprcol1             =$d029

                .eq cram                =$d800
```

```
                .eq jf2                 =$dc00
                .eq ciairq              =$dc0d
                .eq ciactrla            =$dc0e

                .eq home                =$e566
                .eq scroll              =$e968
                .eq bsout               =$ffd2

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

atype           .ts "ClipFull"
routtablo       .by <(dithoff),<(dithhab),<(dithpat),<(dithnoi),<(dithrnd)
routtabhi       .by >(dithoff),>(dithhab),>(dithpat),>(dithnoi),>(dithrnd)
offs            .by $00,$10,$20,$30,$00,$10,$20,$30

; ------------------------------------------------ Event: Toggle Dither Type

ev_dith         ldx gr_dither           ; get current dither type
                lda sc_merk+1           ; where did the mouseclick occur?
                cmp #$0d
                bcs dth0
                dex                     ; left: decrement downto 0
                bpl dth1
                ldx #$04                ; start over at 4
                bne dth1
dth0            inx                     ; increment up to 4
                cpx #$05
                bcc dth1
                ldx #$00                ; start over at 0
dth1            stx gr_dither           ; set dither type
                lda #<(diththype)       ; show new status
                sta sc_texttab
                lda #>(diththype)
                sta sc_texttab+1
                txa
                asl
                tay
                lda dtadr,y
                tax
                lda dtadr+1,y
                tay
                sta gr_redisp
                jmp gd_xtxtggl

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

dtadr           .by <(dtoff),>(dtoff),<(dthab),>(dthab)
                .by <(dtpat),>(dtpat),<(dtnoi),>(dtnoi),<(dtrnd),>(dtrnd)
dtoff           .ts "    Off     @"
dthab           .ts "  Ordered   @"
dtpat           .ts "  Pattern   @"
dtrnd           .ts "   Random   @"
dtnoi           .ts "   Noise    @"

; ------------------------------------------------ Screenlist: Palette Requester
```

```
pallst      .by $00
            .by $04,$02,$24,$10
            .by $91,$00,$00
            .ts "Palette@"
            .by $0a,$03,$09,$03
            .by $d0,<(se_paldis),>(se_paldis)
            .ts "Display@"
pickbox     .by $06,$03,$22,$04
            .by $60,<(se_ppick),>(se_ppick)
pick        .by $0a,$1c,$09,$03
            .by $20,$00,$00
palbox      .by $0d,$03,$06,$04
            .by $60,<(se_pset),>(se_pset)
            .by $11,$03,$09,$03
            .by $dc,<(se_pdeft),>(se_pdeft)
            .ts "Default@"
            .by $11,$0c,$08,$03
            .by $dc,<(se_pundo),>(se_pundo)
            .ts " Undo @"
            .by $11,$14,$08,$03
            .by $dc,<(se_pacc),>(se_pacc)
            .ts "Accept@"
            .by $11,$1c,$09,$03
            .by $dc,<(se_pcanc),>(se_pcanc)
            .ts "Cancel @"
            .by $c0,$09,$0d,$0b
            .ts "Pick Color.@"
            .by $80

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

ev_bal      jsr savescvek           ; save vector to main list
            ldx #<(balreq)          ; show balancing requester
            ldy #>(balreq)
            jsr gd_screen
            jsr gbtab               ; save balancing table for Cancel
            lda gr_brt              ; save brightness and contrast values
            sta brttmp
            lda gr_crt
            sta crttmp
            jsr brt0
            jmp gd_eloop            ; wait for further events

; ------------------------------------------------ Select Gadget Brightness or Contrast

gbset       ldx #$00
            stx sc_loop
            and #$01                ; determine whether even or odd
            bne ic0                 ; thus selecting the appropriate value display
            ldx #<(brtbox)
            ldy #>(brtbox)
            bne bgettab
ic0         ldx #<(ctrbox)
            ldy #>(ctrbox)
```

```
bgettab     stx sc_screentab
            sty sc_screentab+1
            lda #$00
            tax
            tay
bgt0        jsr gd_setpos           ; get 4 parameters
            cpx #$04
            bne bgt0
            jsr gd_trim             ; point to within in the display box
            jmp gd_initmove         ; compute screen address

; ---------------------------------------------- Subevent: Increment Brightness

se_incbrt   lda #$00                ; even function numbers: brightness
            .by $2c

; ---------------------------------------------- Subevent: Decrement Brightness

se_decbrt   lda #$02
            .by $2c

; ---------------------------------------------- Subevent: Increment Contrast

se_incctr   lda #$01                ; odd function numbers: contrast
            .by $2c

; ---------------------------------------------- Subevent: Decrement Contrast

se_decctr   lda #$03
            pha
            jsr gbset               ; compute screen address
            pla
            tay
            and #$01                ; address the appropriate value
            tax
            lda brttmp,x
            cpy #$02                ; determine function: inc or dec
            bcc ic2

            tay                     ; dec: is zero? (minimum)
            beq show                ; yes, no more changes
            dec brttmp,x            ; no, decrease
            tya
            bne show                ; unconditional branch

ic2         tay                     ; inc: is 31? (maximum)
            cpy #$1f
            bcs show                ; yes, no more changes
            inc brttmp,x            ; no, increase

show        lda brttmp,x            ; get value
            ldy #$20                ; sign (positive)
            cmp #$10                ; is at least 16? (median; shows as 0)
            bmi nega                ; no, lower (show as negative values)
```

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

posi        sbc #$10                ; yes, subtract 16
sh0         sty sc_movetab
            jsr gd_xcnvdez          ; convert value to digits
            sta sc_movetab+2        ; ones' digit
            txa                     ; tens' digit
            and #$0f                ; is zero?
            bne sh1
            ldx #$20                ; yes, replace by space
sh1         stx sc_movetab+1
            lda #$03                ; show 3 chars on screen
            sta sc_br
            ldx #$01
            jsr fi1
            jsr makebtab            ; re-compute balancing table

            ldy #$0f                ; convert balancing values to dither patterns
sh2         lda gr_btab,y
            ora #$e0
            sta tpatbox,y           ; write to output buffer
            dey
            bpl sh2
            ldx #<(patbox)          ; compute address of patternbox on screen
            ldy #>(patbox)
            jsr bgettab
            ldy #$06                ; show pattern
            jmp gd_xtxout1

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

nega        sta sc_merk             ; subtract value from 16
            lda #$10
            sec
            sbc sc_merk
            ldy #$2d                ; sign (negative)
            bne sh0                 ; unconditional branch

; ------------------------------------------------ Subevent: Reset Balancing Table

se_balrst   lda #$10                ; set median value
            sta brttmp
            sta crttmp
brt0        lda #$00                ; flag: first pass
brt1        pha
            pha
            jsr gbset               ; reset visible values
            pla
            tax
            jsr show
            pla
            bne brt2                ; finish if second pass
            lda #$01                ; flag: second pass
            bne brt1                ; unconditional branch
brt2        clc
```

```
                rts

; ------------------------------------------------ Subevent: Accept Balancing Values

se_balok        lda brttmp              ; finalize values
                sta gr_brt
                lda crttmp
                sta gr_crt
                lda #$01                ; force re-render
                sta gr_redisp
                bne bq0                 ; leave requester

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

se_balq         ldy #$0f                ; restore original values
bq1             lda btabold,y
                sta gr_btab,y
                dey
                bpl bq1
bq0             sec                     ; leave requester
                rts

; ------------------------------------------------ Save Balancing Table

gbtab           ldy #$0f
gbt0            lda gr_btab,y
                sta btabold,y
                dey
                bpl gbt0
                rts

; ------------------------------------------------ Compute Balancing Table

makebtab        lda crttmp              ; contrast
                asl
                tax
                lda ctrtab,x
                sta ls_temp
                sta sc_pos
                inx
                lda ctrtab,x
                sta ls_temp+1
                sta sc_pos+1
                lda #$00
                sta sc_loop
                asl sc_pos
                rol sc_pos+1
                asl sc_pos
                rol sc_pos+1
                asl sc_pos
                rol sc_pos+1
                lda sc_pos              ; value times 8
                sta sc_texttab
                lda sc_pos+1
                sta sc_texttab+1
```

```
            asl sc_pos
            rol sc_pos+1
            lda sc_pos              ; value times 24
            adc sc_texttab
            sta sc_pos
            lda sc_pos+1
            adc sc_texttab+1
            sta sc_pos+1
            bcc mbskip
            inc sc_loop

mbskip      ldx #$00                ; create 48 entry table
mbloop      lda sc_pos
            sec
            sbc ls_temp
            sta sc_pos
            lda sc_pos+1
            sbc ls_temp+1
            sta sc_pos+1
            lda sc_loop
            sbc #$00
            sta sc_loop
            lda #$07
            sec
            sbc sc_pos+1
            bcc mb0
            pha
            lda #$00
            sbc sc_loop
            pla
            bcs mbskip1
mb0         lda #$00
mbskip1     sta bttab,x
            inx
            cpx #$18
            bne mbloop

mbloop1     lda sc_pos
            clc
            adc ls_temp
            sta sc_pos
            lda sc_pos+1
            adc ls_temp+1
            sta sc_pos+1
            pha
            lda sc_loop
            adc #$00
            sta sc_loop
            pla
            clc
            adc #$07
            bcs mbf
            cmp #$10
            bcc mbskip2
mbf         lda #$0f
```

```
mbskip2     sta bttab,x
            inx
            cpx #$30
            bne mbloop1

            ldx brttmp                  ; brightness
            ldy #$00
mbloop2     lda bttab,x
            sta gr_btab,y
            inx
            iny
            cpy #$10
            bne mbloop2
            rts

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

list        .by $00,$00
bktmp       .by 0
pdis        .by 0
ctrtab      .by $26,$00,$30,$00,$48,$00,$60,$00
            .by $90,$00,$98,$00,$93,$00,$a4,$00
            .by $ab,$00,$b2,$00,$ba,$00,$c3,$00
            .by $cd,$00,$d8,$00,$dc,$00,$f1,$00
            .by $00,$01,$25,$01,$2c,$01,$3b,$01
            .by $55,$01,$74,$01,$9a,$01,$c7,$01
            .by $00,$02,$49,$02,$ab,$02,$33,$03
            .by $00,$04,$55,$05,$74,$06,$00,$08

; ------------------------------------------------ Screenlist for Balancing Requester

balreq      .by $00
            .by $06,$07,$1a,$0c
            .by $91,$00,$00
            .ts "Balancing@"
            .by $08,$08,$03,$03
            .by $c0,<(se_decbrt),>(se_decbrt)
            .ts "<@"
brtbox      .by $08,$0b,$06,$03
            .by $a0,$00,$00
            .ts "   @"
            .by $08,$11,$03,$03
            .by $c0,<(se_incbrt),>(se_incbrt)
            .ts ">@"
            .by $08,$14,$03,$03
            .by $c0,<(se_decctr),>(se_decctr)
            .ts "<@"
ctrbox      .by $08,$17,$06,$03
            .by $a0,$00,$00
            .ts "   @"
            .by $08,$1d,$03,$03
            .by $c0,<(se_incctr),>(se_incctr)
            .by ">@"
patbox      .by $0c,$0b,$12,$03
            .by $a0,$00,$00
```

```
tpatbox     .by $e0,$e1,$e2,$e3,$e4,$e5,$e6,$e7
            .by $e8,$e9,$ea,$eb,$ec,$ed,$ee,$ef,$00
            .by $0f,$08,$08,$03
            .by $dc,<(se_balrst),>(se_balrst)
            .ts "Reset@"
            .by $0f,$10,$08,$03
            .by $dc,<(se_balok),>(se_balok)
            .ts "Accept@"
            .by $0f,$18,$08,$03
            .by $dc,<(se_balq),>(se_balq)
            .ts "Cancel@"
            .by $c0,$0a,$08,$0a
            .ts "Brightness@"
            .by $c0,$0a,$15,$08
            .ts "Contrast@"
            .by $80

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

ev_prviu    jsr gd_initmove         ; save SC_VEKT20
            pha
            lda sc_vekt20
            pha
            jsr gd_invert           ; invert preview area (black)
            ldx #$c0                ; save data, make space for 3 sprites
            ldy #$00
loop1       lda spr2-1,x
            sta dbuf-1,x
            tya
            sta spr2-1,x
            dex
            bne loop1
            ldx #$c0                ; clear 3 more sprites
loop2       sta spr5-1,x
            dex
            bne loop2

            sta sprpri              ; init parameters
            sta xexp
            sta yexp
            lda #$fc                ; multimode sprites
            sta sprmulti
            ldx #$05                ; sprites 2-7: medium gray
            lda #$0c
loop3       sta sprcol1,x
            dex
            bpl loop3
            lda #$0b                ; dark gray to all
            sta sprcol2
            lda #$0f                ; light gray to all
            sta sprcol3

            ldx #$0b                ; set position
loop4       lda xytab,x
            sta sprxy,x
```

```
            dex
            bpl loop4
            lda xmsb            ; x>255?
            ora #$fc
            sta xmsb
            ldx #$05            ; activate sprite definition blocks
            ldy #$ff
ploop       tya
            sta sprptr,x
            dey
            dex
            bpl ploop
            lda #$ff            ; activate all sprites
            sta sprreg

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

            ldx #$02            ; 3 times 12 tiles (times 20 rows)
m0loop      txa
            pha
            lda srcl,x          ; source address (4bit)
            sta sc_pos
            lda srch,x
            sta sc_pos+1
            lda dstl,x          ; destination (sprites)
            sta sc_loop
            lda dsth,x
            sta sc_loop+1
            lda #$00
            sta sc_screentab
            jsr makeit          ; render 1 third of the image
            pla
            tax
            dex
            bpl m0loop          ; three times

            inx                 ; wait for key
            stx sc_taste
wait        lda sc_taste
            beq wait
;
            lda xmsb            ; reset all data
            and #$03
            sta xmsb
            pla
            sta sc_vekt20
            pla
            sta sc_vekt20+1
            jsr gd_invert       ; and change from black to blue
            ldx #$c0
piu0        lda dbuf-1,x
            sta spr2-1,x
            dex
            bne piu0
```

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

spron       lda #$03                ; pointer sprites on
            .by $2c
; ------------------------------------------------
sproff      lda #0                  ; pointer sprites off
            sta sprreg
            clc
            rts

; ------------------------------------------------ Render image to sprites

makeit      ldx #$00                ; index, must be zero
            ldy #$14                ; counter for tile rows (only 20)

prloop      tya                     ; save it
            pha
            lda #$40                ; flag: three passes (value gets shifted left)
            sta ls_temp+1

dzloop      lda #$03
            sta brttmp

zloop       ldy #$60                ; cover 4 tiles
            lda ls_temp+1           ; second pass: indent to 4 pixels below
            bpl bloop
            ldy #$70

bloop       lda (sc_pos),y          ; get left 4b pixel
            lsr
            lsr
            lsr
            ror crttmp              ; get 2 pixels
            lsr
            ror crttmp
            tya                     ; next tile
            sec
            sbc #$20
            tay
            bpl bloop

            lda crttmp              ; write to sprite
            sta (sc_loop,x)
skip64      inc sc_loop             ; proceed in destination
            bne s1
            inc sc_loop+1
s1          inc sc_screentab        ; count bytes in sprite
            lda sc_screentab
            cmp #$3f                ; skip byte 64 in sprite
            beq skip64

            lda sc_pos              ; add 4 tiles (128)
            clc
            adc #$80
            sta sc_pos
```

```
                bcc skip
                inc sc_pos+1
skip            ror brttmp              ; second and third pass (3 bytes width in a sprite)
                bcs zloop

                lda sc_pos              ; back 3 times 4 tiles
                sec
                sbc #$80
                sta sc_pos
                lda sc_pos+1
                sbc #$01
                sta sc_pos+1
                rol ls_temp+1           ; first/second pass: loop
                bcc dzloop

                lda sc_pos              ; add 1280 (1 tile row)
                clc                     ; (could be optimized...)
                adc #$00
                sta sc_pos
                lda sc_pos+1
                adc #$05
                sta sc_pos+1

                pla                     ; decrement tile row counter
                tay
                dey
                bpl prloop              ; 20 times?

                rts                     ; yes, finished

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

src             .by $40,$c0,$40         ; start of scan addresses
srch            .by $4d,$4b,$4a
dstl            .by $80,$00,$80         ; destination sprite addresses
dsth            .by $3f,$3f,$3e
xytab           .by $07,$91,$07,$a6
                .by $1f,$91,$1f,$a6
                .by $37,$91,$37,$a6
                .by $00

; ------------------------------------------------ Event: GoDot Guru

ev_info         jsr sproff              ; pointer off

                lda #$05                ; scroll screen 5 rows down
                sta sc_screentab
scrloop         jsr home                ; be sure to be at home position
                ldx #$00
                jsr scroll              ; scroll screen down
                dec sc_screentab
                bne scrloop

                lda border              ; save current main color
                sta crttmp
```

```
                lda charreg             ; save $d018
                sta sc_screentab+1

                lda #$00                ; clear keyboard
                sta ls_temp
                sta gd_tbuffer
                jsr set                 ; activate raster irq
                jsr text                ; show Guru text ("GoDot Creation Nr...")
tastean         lda gd_tbuffer          ; wait for key
                bne ok                  ; key pressed, finish
                jsr blink               ; otherwise: blink box
                lda jf2                 ; fire on joystick?
                and #$10
                bne tastean             ; no, wait

ok              jsr off                 ; de-activate Guru
                jsr spron               ; pointer on
                sec                     ; leave to main screen
                rts

; ------------------------------------------------ Guru IRQ

set             sei
                lda irq
                sta oldirq
                lda irq+1
                sta oldirq+1
                lda #<(newirq)
                sta irq
                lda #>(newirq)
                sta irq+1
                lda #$00
                sta raster
                lda raster-1
                and #$7f
                sta raster-1
                lda #$81
                sta rirqen
                lda #$80
                sta ciactrla
                cli
                rts

; ------------------------------------------------ New IRQ

newirq          sei
                lda reqirq
                sta reqirq
                bmi rasterirq
                lda ciairq
                cli
                jmp gd_endirq

; ------------------------------------------------ Guru Activity
```

```
rasterirq       sei
                lda raster
                bne gskip
                lda #$00                ; black screen (above raster split)
                sta border
                sta backgr
                lda #$17                ; standard charset
                sta charreg
                lda #$54
                sta raster
                lda raster-1
                and #$7f
                sta raster-1
                cli
                jmp (oldirq)

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

gskip           ldx #$09
irqwait         nop
                dex
                bne irqwait
                lda crttmp              ; blue screen (below raster split)
                sta border
                sta backgr
                lda sc_screentab+1      ; GoDot charset
                sta charreg
                lda #$00
                sta raster
                lda raster-1
                and #$7f
                sta raster-1
                cli
                jmp gd_endirq

; ------------------------------------------------ End Guru

off             sei
                lda #$93                ; clear screen (and home)
                jsr bsout
                lda #$00                ; reset all values
                sta rirqen
                lda oldirq
                sta irq
                lda oldirq+1
                sta irq+1
                lda #$01
                sta ciactrla
                lda crttmp
                sta border
                sta backgr
                lda sc_screentab+1
                sta charreg
                cli
                rts
```

```
; ------------------------------------------------ Create Guru Information Message

text        ldx #2
            ldy #3
tx03        lda gd_version-1,x      ; create version number digits
            pha
            and #15
            ora #$30
            sta vermain,y
            pla
            lsr
            lsr
            lsr
            lsr
            ora #$30
            dey
            sta vermain,y
            dey
            dex
            bne tx03

            iny
tx02        lda created,y           ; create version number string
            cmp #$2e
            beq tx00
            sta versn,x
            inx
tx00        iny
            cpx #$03
            bne tx01
            inx
tx01        cpx #$0a
            bne tx02

            lda #<(gurubox)         ; write box border
            sta ls_temp+1
            lda #>(gurubox)
            sta brttmp
            ldx #$00
            stx sc_zl
            stx sc_sp
            stx sc_loop
            ldx #$04
            stx sc_ho
            ldx #$28
            stx sc_br
            jsr gd_initmove
            ldy #$08
            sty sc_merk
            jsr makeline
            jsr gd_backu
            jsr gd_plus40
            jsr makeline
            ldx sc_ho
```

```
                dex
                jsr fi2
                jsr makeline
                jsr gd_backu

                ldy #$00                ; write text (light red: $0a)
                ldx #$0a
                stx sc_chgcol
txloop          lda line1,y
                sta scr1,y
                txa
                sta scr1+$d400,y
                iny
                cpy breit1
                bne txloop
                ldy #$00
txloop2         lda line2,y
                sta scr2,y
                txa
                sta scr2+$d400,y
                iny
                cpy breit2
                bne txloop2
                rts                     ; finished

; ---------------------------------------------- Box data

gurubox         .by $7d,$40,$6d,$5d,$20,$5d,$6e,$40
                .by $70
line1           .ts "A.Dettke & W.Kling"
line2           .ts "GoDot Creation Nr. "
vermain         .ts "00000"
versn           .ts "000.00000000"
breit1          .by $12
breit2          .by $24

; ---------------------------------------------- Blink Box Border

blink           ldy #$28
                lda sc_chgcol           ; write color
bl0             sta cram,y
                sta cram+119,y
                dey
                bpl bl0
                sta cram+79
                sta cram+80
                ldx #$fa                ; wait a bit
                jsr delay2
                lda ls_temp             ; toggle flag
                eor #$01
                sta ls_temp
                bne bl1
                lda #$0a                ; blink between light red...
                .by $2c
bl1             lda #$00                ; and black (background color)
```

```
                sta sc_chgcol
                rts

; ------------------------------------------------
modend      .en                         ; end of upmem routines
```