godotlib.lib

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

```
; --------------------------
;
; Library of GoDot System Variables & Routines
;
; --------------------------

ls_len          = $1c       ; length of filename for savers/filecopy
sy_oldposx      = $1d       ; mouse, can't be altered
sy_oldposy      = $1e       ; mouse, can't be altered
ls_loadsave     = $1f       ; flag: SAVE if 0, LOAD if not
sc_vekt20       = $20       ; /$21, pointer to screen address
sc_descriptor   = $22       ; Gadget descriptor:
sc_zl           = $22       ;   row in tiles
sc_sp           = $23       ;   column
sc_br           = $24       ;   width
sc_ho           = $25       ;   height
ls_lines        = $26       ; amount of files currently in filerequester
rm_reusize      = $27       ; encoded size of attached REU (none if 0, see section "values"
below))
ls_found        = $28       ; flag: files of current type existing

;               $29 to $2f: reserved
;               $30 up to $53: free for programmer

sy_soffx        = $7a       ; x-offset of mouse pointer to its hotspot
sy_soffy        = $7b       ; y-offset dito

sc_pos          = $a4       ; /$a5, variously usable vector
sc_loop         = $a6       ; flag: target address textram if 0, colorram if not
ls_temp         = $a7       ; variously usable
ls_vekta8       = $a8       ; /$a9, variously usable, mainly used as 16 bit counter
gr_nr           = $aa       ; contains number of function for render routines
sc_screentab    = $ab       ; /$ac, pointer to screenlists or gadget descriptors
gd_modswitch    = $ad       ; stores highbyte of modules when exchanging them
sc_texttab      = $ae       ; /$af, pointer for text outputs
sc_merk         = $b0       ; /$b1, variously usable
ls_adrmisc      = $b2       ; /$b3, target load address when LS_DIRMASK set to 8
sc_flag         = $b4       ; text appearance flag

;               $b5/$b6: unused

sc_keyprs       = $c6       ; key currently pressed

;               $f7 up to $ff: unused (except $ff e.g. in mod..FileCopy)

sy_numbers      = $0100     ; space for converting bytes to digits (16 bytes)
ls_nambuf       = $0200     ; buffer for filenames (16 bytes)

sy_area1        = $0220     ; 46 bytes temporarily used by ldr..SnapShot
```

```
rm_sram         = $0254     ; size of SuperRAM
ls_cmdtypes     = $0255     ; drive types for attached units if CMD drives ("H", "F", or "R")
sy_tbuffer      = $0277     ; buffer for keys pressed
sy_area2        = $0293     ; 11 bytes temporarily used by ldr.GIF
gr_ctr          = $02a7     ; current value of contrast
gr_brt          = $02a8     ; current value of brightness
rm_nextram      = $02a9     ; next free directory entry in ram device (number)
rm_rlen         = $02aa     ; length of filename for ram device
rm_ramundo      = $02ab     ; flag: undo performed (0 if not)
ls_bootdrive    = $02ac     ; drive where GoDot booted from
rm_cntlns       = $02ad     ; amount of files in file requester (ram device)
gr_qlock        = $02ae     ; flag: quantization locked if 0 (unused yet)
rm_tmpsaved     = $02af     ; flag: temp area filled
gr_basepal      = $02b0     ; base palette (16 bytes)
sc_pointer      = $02c0     ; definition of mouse pointer (63 bytes)

sc_shadow       = $0334     ; gadget colors: right and bottom edges
sc_light        = $0335     ; left and top edges
sc_normtext     = $0336     ; textcolor
sc_hilite       = $0337     ; highlighted text color
sc_chgcol       = $0338     ; used to exchange colors

ls_showfiles    = $0339     ; flag: show *all* files in filerequester if not 0
rm_combuf       = $033a     ; carries REU command (dev.REU)
sc_iflag        = $033b     ; flag: amount of permitted input chars (-1) if not 0
sc_lastclpzl    = $033c     ; row of last clip
sc_lastclpsp    = $033d     ; column
sc_lastclpbr    = $033e     ; width
sc_lastclpho    = $033f     ; height

gr_orderedpat   = $0340     ; definition of Ordered Pattern, 64 bytes
gr_rgb          = $0380     ; definition of C64 colors, 48 bytes
gr_btab         = $03b0     ; table of balancing (16 bytes)
gr_defpals      = $03c0     ; definition of the 14 different default palettes (28 bytes)
ls_lastname     = $03dc     ; name of current picture in buffer, 16 bytes
rm_swapw        = $03ec     ; buffer for swap values (ram device, 7 bytes)
rm_werte        = $03f3     ; buffer for REU register values (ram device, 7 bytes)

ls_sysdrive     = $03fa     ; current system drive (bootdrive or unit ram)
rm_ramvec       = $03fd     ; vector to ram device (contains "JMP $cab0" if active)
sy_versioninit  = $03fb     ; /$03fc: version of GoDot Launcher (in decimal mode)

gr_qtab         = $0ee0     ; table of quantization, 16 bytes
sc_undochar     = $0ef8     ; definition of multipurpose char 223, 8 bytes
gr_pattern      = $0f00     ; definition of System Pattern, 128 bytes

sc_screenvek    = $0f88     ; /$0f89: vector to current screenlist
sy_event        = $0f8a     ; /$0f8b: vector to module's current event routine
sy_irqalt       = $0f8c     ; /$0f8d: vector to pre-GoDot IRQ

sc_maincolor    = $0f8e     ; color of border and background
rm_ramfunc      = $0f8f     ; number of current ram function
rm_nextentry    = $0f90     ; next directory entry in ram device (number)
sy_newposx      = $0f91     ; mouse, can't be altered
sy_newposy      = $0f92     ; mouse, can't be altered
```

```
sc_scvek2       = $0f93     ; /$0f94: buffer for SC_SCREENVEK
gr_cmode        = $0f95     ; current graphics mode (0 = hires; 2 = multicolor)
gr_howmany      = $0f96     ; current amount of grays to be rendered (2 to 16)
gr_picked       = $0f97     ; current palette color (msb set if inactive)
gr_palette      = $0f98     ; current palette (16 bytes)

ls_drive        = $0fa8     ; current drive (8 - 12, 5 bytes)
ls_dirmask      = $0fa9     ; type of file/data to be loaded (see "values" section below)
ls_units        = $0faa     ; flags: drivetype and presence (5 bytes)
ls_ramdrive     = $0fae     ; last of LS_UNITS is flag for RAM ("drive" 12)
ls_track        = $0faf     ; current track on disk
ls_sector       = $0fb0     ; current sector
ls_index        = $0fb1     ; offset into current dirblock
sy_framecount   = $0fb2     ; /$0fb3: not used yet
ls_err1         = $0fb4     ; first digit of floppy error message (unused but always set)
ls_err2         = $0fb5     ; second digit
ls_cblocks      = $0fb6     ; amount of directory blocks in filerequester
sc_clicked      = $0fb7     ; counter for doubleclick
sc_ticks        = $0fb8     ; starting value of SC_CLICKED
ls_flen         = $0fb9     ; length of filename
ls_flen2        = $0fba     ; length of filename without signature (ls_flen-4)
ls_first        = $0fbb     ; first directory sector

gr_dither       = $0fbc     ; current dither type
gr_redisp       = $0fbd     ; flag: graphics must be rendered on "Display" (if not 0)
sc_clipzl       = $0fbe     ; row of current clip
sc_clipsp       = $0fbf     ;   column
sc_clipbr       = $0fc0     ;   width
sc_clipho       = $0fc1     ;   height
sc_clipped      = $0fc2     ; flag: clip set and active if not 0
ls_saveto       = $0fc3     ; drive where currently to save to
ls_loadfrom     = $0fc4     ; drive where currently to load from
sc_stop         = $0fc5     ; flag: STOP pressed if not 0 (must be initialized by 0)
gr_bkcol        = $0fc6     ; background color for multicolor mode
rm_vdcflag      = $0fc7     ; status of VDC ram
sc_movetab      = $0fd8     ; text/textcolor output buffer, 39 bytes plus $00 (40 bytes)

; -------------------------- System Subroutines/System Tables

sy_boxtab       = $1000     ; table of definition strings for gadget boxes
                            ; (144 bytes, 9 bytes each boxtype)
gd_initmove     = $1090     ; calculate screen address out of SC_ZL, SC_SP and SC_LOOP
gd_mal40        = $10c1     ; SC_POS times 40
gd_mal20        = $10c5     ; SC_POS times 20
gd_mal10        = $10c9     ; SC_POS times 10
gd_mal5         = $10cd     ; SC_POS times 5
gd_plus40       = $10e7     ; SC_VEKT20 plus 40
gd_backu        = $10f5     ; output SC_MOVETAB on screen at SC_VEKT20, length in
SC_BR
gd_blank        = $1102     ; fill SC_MOVETAB with Space, amount in .Y
gd_bl3          = $1105     ; fill SC_MOVETAB with byte in .A, amount (-1) in .Y
gd_bl4          = $1108     ; fill SC_MOVETAB with byte in .A, amount in .Y
gd_fcol         = $110c     ; output color square on screen, offset to color in .X (from
SC_SHADOW)
```

```
gd_fi0          = $1114     ; output char square on screen, char in .A (eg. a Space); after
GD_INITMOVE
gd_fi1          = $111b     ; output SC_MOVETAB on screen, height in .X; after
GD_INITMOVE
gd_invert       = $1125     ; revert square on screen, width and height in SC_BR and
SC_HO; after GD_INITMOVE
gd_box          = $1141     ; output gadget box on screen, type in .X, appearance in
SC_FLAGS, size in SC_DESCRIPTOR
gd_setpos       = $11cc     ; retrieve gadget parameters from screenlist and write to
SC_DESCRIPTOR
gd_screen       = $11e0     ; output GoDot requester, vector to screenlist in .X and .Y
(lo/hi)
gd_text         = $1245     ; output text to screen, vector to text in SC_SCREENTAB,
offset (-1)  in .Y
gd_position     = $1289     ; recompute pointer position to row (SC_MERK) and column
(SC_MERK+1)
gd_evntsuch     = $12ad     ; on click: search screenlist for event routine at this screen
location
gd_esm3         = $1303     ; same as GD_INVERT but computing of screen address
included
gd_delay        = $1319     ; slowdown for 4 ms (simple counter of cycles, not beam
driven)
gd_dl2          = $131b     ; slowdown for .X ms
gd_trim         = $1324     ; SC_ZL and SC_SP plus 1, SC_BR and SC_HO minus 2
gd_irq          = $1331     ; GoDot's IRQ routine
gd_endirq       = $13e5     ; IRQ tail, to be used if toggled off for any reason
gd_endnmi       = $13ea     ; NMI tail

sy_version      = $1415     ; kernel version number in decimal mode
sy_into         = $1417     ; entrance vector to GoDot
gd_xmloop       = $1420     ; external JMP to MLOOP ("main loop")
gd_eloop        = $1423     ; event loop routine, SC_SCREENVEK points to current
screenlist
gd_savescvek    = $143f     ; save SC_SCREENVEK to SC_SAVESCVEK

ev_exit         = $144c     ; event routine for main screen function "Exit"
se_cancel       = $1453     ; subevent "Cancel"
se_ende         = $1455     ; subevent "Leave GoDot"

gd_xopen        = $1458     ; mouse pointer off, OPEN channels 13 and 15 for input
gd_xclose       = $145b     ; CLOSE 13 and 15
gd_xmess        = $145e     ; output floppy error message at row 23, column 4 (message &
gauge bar area)
gd_xtxout1      = $1461     ; output text, pointer to textparameters in SC_SCREENTAB,
offset (-1) in .Y
gd_xtxout2      = $1464     ; output text to current SC_VEKT20, pointer to text in .X and
.Y (lo/hi)
gd_xtxout3      = $1467     ; output textbuffer (SC_MOVETAB) to current SC_VEKT20
gd_xinput       = $146a     ; input text (parameters set by relating gadget, needs any
saver)
gd_xtxtggl      = $146d     ; swap gadget text and an alternate one, vector to that in .X
and .Y (lo/hi)
gd_xswap4k      = $1470     ; swap Execution Area and module area, highbyte of specific
module area in .X
gd_xcnvdez      = $1473     ; convert value in .A to 2 digits (in .X and .A)
```

```
gd_xloadm       = $1476     ; open file requester for non-system filetypes,
gd_listen       = $147a     ; sends Listen (drive in .A) and secondary address after Listen
(in .Y)
gd_talk         = $1481     ; sends Talk (drive in .A) and secondary address for Talk (in .Y)
gd_sendcom      = $1488     ; sends floppy command, address in .X and .Y (lo/hi), length in
.A, then Unlisten
gd_testdrive    = $14c9     ; checks drive (in .A) for presence, if not returns Carry flag
*not* set
gd_setmess      = $14dc     ; set position parameters for row 23, column 4 (needed by
GD_INITMOVE)
gd_setpar       = $14e2     ; set own parameters, row in .A, column in .X, width in .Y
(height is 1)
gd_clrms        = $14f0     ; clear status line on screen (row 23, column 4)
gd_clrline      = $14f3     ; clear current line (set by GD_SETPAR)
gd_error1       = $14fe     ; output "Off." on status line
gd_which        = $1509     ; check all drives for presence

se_scdwn        = $151e     ; subevent "Scroll Down" (in any file requester)
se_scup         = $1526     ; subevent "Scroll Up"

gd_prepdir      = $1560     ; clear directory window and input gadget
gd_sun2         = $1562     ; clear directory window (.A = 1)
gd_clickon      = $1588     ; activate tick counter (SC_CLICKED) for double click

se_select       = $158f     ; subevent "Select File"

gd_cnvasc       = $15fd     ; convert LS_NAMBUF (screencode) to petscii

se_units        = $1621     ; subevent "Select Drive"
se_delete       = $16d0     ; subevent "Scratch File"
ev_exec         = $1742     ; event "Execute Module"
se_save         = $1746     ; subevent "Save Data"
se_input        = $174d     ; subevent "Input Filename" (part of file requester)
ev_disp2        = $1753     ; event "Display Rendered Graphics"
ev_prviu        = $1756     ; event "Show Preview"
ev_info         = $1759     ; event "Show Guru Alert"
ev_area         = $175c     ; event "Toggle Clip/Full"
ev_dith         = $175f     ; event "Select Dither Type"
ev_display      = $1762     ; event "Render Graphics"
ev_pal          = $1765     ; event "Open Palette Requester"
ev_balance      = $1768     ; event "Open Balancing Requester"

gd_getback      = $1777     ; reswap GD_XSWAP4K

ev_sccont       = $177f     ; event "Toggle Graphics Mode"

gd_makeqtab     = $17c9     ; compute table of quantization (GR_QTAB)

ev_cols         = $17e0     ; event "Select Amount of Colors/Grays"
se_load         = $181b     ; subevent "Load/Save File", according to LS_LOADSAVE
se_ldcan        = $18f0     ; subevent "Cancel Load/Save"
ev_load         = $1918     ; event "Load a Graphics File"
ev_save         = $1922     ; event "Save a Graphics File"

gd_xload        = $1924     ; Load with DM_MISC set (any data file, care for LS_ADRMISC!)
```

```
ev_ldr          = $192e     ; event "Load a Loader Module"
ev_svr          = $1938     ; event "Load a Saver Module"
ev_mod          = $1942     ; event "Load an Image Processing Module" (a "Modifier")

gd_xloadsave    = $194b     ; open filerequester
gd_setspos      = $1a17     ; set position parameters for Units gadget in filerequester
gd_showdrv      = $1a2a     ; output drive types in Units gadget
gd_getdir       = $1aad     ; read one diskblock, track in LS_TRACK, sector in LS_SECTOR
gd_send         = $1ad5     ; send U1-command to drive, LS_TRACK and LS_SECTOR have
to be specified

gd_dir          = $1b3f     ; GoDot show directory routine
ls_ftabt        = $1b49     ; table of 4 current starting tracks (of attached drives)
ls_ftabs        = $1b4d     ; table of 4 current starting sectors (of attached drives)
gd_makedir      = $1b94     ; produce a (max.) 16 entry directory on screen
gd_cnvbc        = $1c5e     ; convert petscii to screen code (at SC_MOVETAB)
gd_sproff       = $1c8d     ; switch Mouse Pointer off
gd_spron        = $1c90     ; switch Mouse Pointer on
gd_onebyte      = $1c96     ; read one byte from file (return in .A) and check status (in .X)
gd_testram      = $1ca9     ; check for active RAM device (status bit must be EQUAL then)
gd_swapd        = $1caf     ; switch RAM device on
gd_reu          = $1cb4     ; send REU command, command in .A, register parameters in
RM_WERTE
ls_picname      = $1fa8     ; name of current image in screencode (16 bytes)
ls_iloader      = ls_picname+$16        ; loader name
ls_imode        = ls_picname+$22        ; graphics mode
ls_idrive       = ls_picname+$2e        ; GoDot mode

sy_global       = $3f40     ; buffer for various purposes (192 bytes)
sy_4bit         = $4000     ; start of 4bit image data
sy_bigbuffer    = $bd00     ; buffer for various purposes (768 bytes)

; -------------------------- ROM Routines/Hardware Registers

intout          = $bdcd     ; convert word value to digits

spritex         = $d002     ; mouse Pointer x-position
spritey         = $d003     ; mouse Pointer y-position
spritehi        = $d010     ; mouse Pointer high byte of x-position

potx            = $d419     ; mouse move x register
poty            = $d41a     ; mouse move y register

joy             = $dc00     ; joystick status register
reubase         = $df00     ; base address of REU
reucom          = $df01     ; command register of REU

irqend          = $ea7e     ; standard finish of IRQ
reset           = $fce2     ; reset vector

checkkeyb       = $ff9f     ; vector: check keyboard
filpar          = $ffba     ; set file parameters
filnam          = $ffbd     ; set filename parameters
copen           = $ffc0     ; open
```

```
close           = $ffc3     ; close
chkin           = $ffc6     ; set input device
ckout           = $ffc9     ; set output device
clrch           = $ffcc     ; clear channel
basin           = $ffcf     ; retrieve one byte
bsout           = $ffd2     ; send one byte




; --------------------------
; ---------- values
; --------------------------

; dm_graphics   = 0         ; dirmask: graphics data (destination $4000)
; dm_ld         = 1         ; loader data (dest. $c000)
; dm_sv         = 2         ; saver data (dest. $d000)
; dm_mod        = 4         ; module data (dest. $e000)
; dm_misc       = 8         ; miscellaneous data (dest. in LS_ADRMISC)

; rs_128k       = 3         ; Size of REU is 128KB
; rs_256k       = 4         ; ... is 256KB
; rs_512k       = 5         ; ... is 512KB
; rs_1m         = 6         ; ... is 1 Meg
; rs_15m        = 7         ; ... is 1.5 Meg
; rs_2m         = 8         ; ... is 2 Meg
; rs_4m         = 9         ; ... is 4 Meg
```