```
;C64 screen and sprite memory addresses
enable_sprite       = $D015
sprite0_mem_pointer = $07F8
sprite1_mem_pointer = $07F9
sprite2_mem_pointer = $07FA
sprite_enable_mc    = $D01C
sprite_expand_x     = $D01D
sprite_collision    = $D01E
;sprite_multicolor0  = $D025
;sprite_multicolor1  = $D026
sprite0_color       = $D027
sprite1_color       = $D028
sprite0_x           = $D000
sprite0_y           = $D001
sprite1_x           = $D002
sprite1_y           = $D003
;sprite2_x           = $D004
;sprite2_y           = $D005
msb_x               = $D010
backgrnd_color_addr = $D021
boarder_color_addr  = $D020
raster_line         = $D012
clear_screen        = $E544
;joystick1           = $DC01 ;56321
joystick2           = $DC00 ;56320

;SID sound chip memory addresses
frelo1   = $d400      ;voice 1 frequency register lo byte
frehi1   = $d401      ;voice 1 frequency register hi byte
vcreg1   = $d404      ;voice 1 control register
atdcy1   = $d405      ;voice 1 attack / decay register
surel1   = $d406      ;voice 1 sustain / release register
frelo2   = $d407      ;so+7
sigvol   = $d418      ;so+24

;status_register     = $030F

;these are memory addresses for the variables starting at 679
;ball_x       = 679
;ball_y       = 681
dir_x        = 682
dir_y        = 683
ball_counter = 684
ball_cnt_label_scr_addr  = 1026  ; 1024+X+(40*Y)  ;1985
ball_cnt_label_color_mem = 55298 ;55296+X+(40*Y)  ;56257
ball_counter_scr_addr = 1032 ;label start addr + label length +
1 ;1991
```

```
;constants
left                = #24  ;left border
top                 = #50  ;top
bottom              = #243 ;bottom
right               = #81  ;right NOTE: 81 with bit set true is
256+81= 337
ball_start_x        = #184
ball_start_y        = #125
ball_color          = #1 ;white ball
paddle_start_x      = #171
paddle_start_y      = #215
paddle_color        = #6 ;blue paddle
backgrnd_color      = #0 ;default darkblue is 6 , black=0
boarder_color       = #15 ;default light blue is 14, light
gray=15

; 10 SYS (49152)
*=$801
        byte
$0E,$08,$0A,$00,$9E,$20,$28,$34,$39,$31,$35,$32,$29,$00,$00,$00

;sprite data
* = 16256;16192
        ;paddle 12x4 top left corner
        byte
$ff,$f8,$00,$ff,$f8,$00,$ff,$f8,$00,$ff,$f8,$00,$00,$00,$00,$00
        byte
$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        byte
$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        byte
$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$05
        ;small 8x8 ball top left corner
        byte
$38,$00,$00,$7c,$00,$00,$fe,$00,$00,$fe,$00,$00,$fe,$00,$00,$7c
        byte
$00,$00,$38,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        byte
$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        byte
$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$01

* = 49152 ;$C000-$CFFF, 49152-53247 Upper RAM area (4096 bytes).
        ;set sprite memory pointers
        lda  #255
        sta  sprite0_mem_pointer ;sprite0 is the ball
```

```
lda  #254
sta  sprite1_mem_pointer ;sprite1 is the paddle
;enable sprites
lda #3
sta enable_sprite       ;enable sprites 0 and 1
;enable sprite multi-color
lda #0 ;zero to disable all multi-color
sta sprite_enable_mc
;set sprite colors
;lda #0
;sta sprite_multicolor0 ;black / shadow
;lda #1
;sta sprite_multicolor1 ;white / light
lda ball_color
sta sprite0_color
lda paddle_color
sta sprite1_color

;sprite to foreground display priority register
;lda #3
;sta 53275

;sprite expand
lda #2
sta sprite_expand_x ;x-expanded #$d01d

;initialize ball direction vars
lda #0
sta dir_x ;set x direction
lda #1
sta dir_y ;set y direction

;set initial ball position
jsr reset_ball

;set initial paddle position
lda paddle_start_x
sta sprite1_x
lda paddle_start_y
sta sprite1_y

;init_screen
lda backgrnd_color
sta backgrnd_color_addr    ;set background color
lda boarder_color
sta boarder_color_addr       ;set border color
;clear screen
```

```
        jsr clear_screen
        jsr clear_sound
        ;setup ballcounter
        lda #49 ;init ballcount to one in ascii
        sta ball_counter
        jsr display_ball_counter_label ;"BALL:"
        jsr display_ball_counter

;wait for raster scan line to be off screen (>250)
raster
        lda raster_line ;$D012
        cmp #250
        bne raster
main
        jsr move_ball_horizontally
        jsr move_ball_vertically
        jsr move_paddle
        jsr check_ball_paddle_collision
        jsr check_ball_wall_collisions
        jmp raster
;END of MAIN

check_ball_wall_collisions
        ;check right wall collisions
        lda msb_x
        and #1
        cmp #1
        bne check_left_ball_wall_collision
        ;ball msb is set
        lda sprite0_x
        cmp right
        beq set_ball_direction_left
check_left_ball_wall_collision
        lda msb_x
        and #1
        cmp #0
        bne raster ;only check left wall when msb is zero, else
exit
        lda sprite0_x
        cmp left
        beq set_ball_direction_right
        rts

set_ball_direction_right
        lda #1
        sta dir_x
        jsr sound
```

```
        rts
set_ball_direction_left
        lda #0
        sta dir_x
        jsr sound
        rts

move_ball_horizontally
        lda dir_x
        cmp #0
        beq move_ball_left
        cmp #1
        beq move_ball_right
        rts

move_ball_right
        inc sprite0_x
        bne dont_toggle_ball_msb_right ;checks zero flag
        lda msb_x
        eor #$01 ;sprite0 x-axis msb
        sta msb_x
dont_toggle_ball_msb_right
        rts

move_ball_left
        lda sprite0_x
        bne dont_toggle_ball_msb_left ;checks zero flag
        lda msb_x
        eor #$01 ;sprite0 x-axis msb
        sta msb_x
dont_toggle_ball_msb_left
        dec sprite0_x
        rts

move_ball_vertically
        lda dir_y
        cmp #0
        beq moveball_up
        cmp #1
        beq moveball_down
        rts

moveball_down
        inc sprite0_y
        jsr check_ball_floor_collision
        rts
moveball_up
```

```
        dec sprite0_y
        jsr check__ball_ceiling_collision
        rts

check__ball_ceiling_collision
        lda sprite0_y
        cmp top
        beq set_ball_direction_down
        rts

check_ball_floor_collision
        lda sprite0_y
        cmp bottom
        beq reset_ball
        rts

set_ball_direction_down
        lda #1
        sta dir_y
        jsr sound
        rts

set_ball_direction_up
        lda #0
        sta dir_y
        jsr sound
        rts

reset_ball
        ;reset ball x,y
        lda ball_start_x
        sta sprite0_x
        lda ball_start_y
        sta sprite0_y
        ;clear ball msb if needed
        lda msb_x
        and #1
        cmp #1
        bne skip_reset_msb
        lda msb_x
        eor #$01
        sta msb_x
skip_reset_msb
        ;increment ball count
        inc ball_counter
        ;roll over digit
        lda ball_counter
```

```
        cmp #58 ;58 in ascii is 9 +1
        beq reset_ball_counter
        ;update ball count display
        jsr display_ball_counter
        rts

reset_ball_counter
        lda #48 ;zero in ascii
        sta ball_counter
        ;update ball count display
        jsr display_ball_counter
        rts

move_paddle_right
        lda msb_x
        and #2
        cmp #2
        bne mv_pad_r
        ;ball msb is set
        lda sprite1_x
        cmp #62 ;RIGHT offset with the paddle width minus ball
width
                ;NOTE: currently RIGHT is already offset with
ball width
                ;assumes paddle is x-axis expanded
        beq dont_toggle_paddle_msb_right
mv_pad_r
        inc sprite1_x
        bne dont_toggle_paddle_msb_right
        lda msb_x
        eor #$02 ;sprite1 x-axis msb
        sta msb_x
dont_toggle_paddle_msb_right
        rts

move_paddle_left
        lda sprite1_x
        bne dont_toggle_paddle_msb_left
        lda msb_x
        eor #$02 ;sprite1 x-axis msb
        sta msb_x
dont_toggle_paddle_msb_left
        lda msb_x
        and #2
        cmp #2
        beq mv_pad_l
        lda sprite1_x
```

```
        cmp left
        bne mv_pad_l
        rts
mv_pad_l
        dec sprite1_x
        rts

move_paddle
        ;read input from joystick
        lda  joystick2    ;joy port 2 $DC00
        and  #8 ;right
        beq move_paddle_right
        lda  joystick2
        and  #4 ;#left
        beq move_paddle_left
        rts

check_ball_paddle_collision
        lda sprite_collision ;53278
        and #2
        cmp #2
        beq ball_paddle_collision
        rts

ball_paddle_collision
        lda sprite_collision
        and #2
        sta sprite_collision ;clear sprite collision bit
        jsr set_ball_direction_up
        rts

display_ball_counter_label ;"BALL:"
        ldx #0
read_ball_cnt_label
        lda ball_counter_label,x
        sta ball_cnt_label_scr_addr,x
        lda #7 ;yellow = 7
        sta ball_cnt_label_color_mem,x
        inx
        cpx #5 ;label text length
        bne read_ball_cnt_label
        sta ball_cnt_label_color_mem,x+1 ;color the counter 2
char right of label
        sta ball_cnt_label_color_mem,x+2
        rts

display_ball_counter
```

```
        lda ball_counter
        sta ball_counter_scr_addr
        rts

sound
        jsr clear_sound
        lda #5
        sta atdcy1      ;s0+5
        lda #5
        sta surel1      ;so+6
        lda #15
        sta sigvol      ;so+24
        lda #7
        sta frelo1      ;so
        lda #27
        sta frehi1      ;so+1
        lda #33
        sta vcreg1      ;so+4
        rts

clear_sound
       ldy #23
clr
       lda #0
       sta frelo1,y
       dey
       bne clr
       rts

;score_label byte 32,19,3,15,18,5,58
ball_counter_label byte 2,1,12,12,58
```