"

```
--------------------------------------------------------------------------------
Phoenix memory map

  0000-3fff  16Kb Program ROM
  4000-43ff  1Kb Video RAM Charset A (4340-43ff variables)  O(x,y) linksboven visueel
                      -433f FOREGROUND charsetA 32Hx26W  90'rotatie dus rechtsboven $4000
                       linksboven ($4320) linksonder ($433f) rechtsonder $401f
  4400-47ff  1Kb Work RAM
  4800-4bff  1Kb Video RAM Charset B (4b40-4bff variables)
                       -4b3f BACKGROUND charsetB  idem maar dan +$0800 dus rb is $4800
                        linksboven ($4b20) linksonder ($4b3f) rechtsonder $481f
      $4BFF  Stack Pointer
  4c00-4fff  1Kb Work RAM
  5000-53ff  1Kb Video Control write-only (mirrored)
             bit#0 determines palette 0 of 1   bit 0 = Player 1 or 2 RAM bank switching
                                               bit 1 = Color palette high bit (6)
  5400-57ff  1Kb Work RAM
  5800-5bff  1Kb Video Scroll Register (mirrored)
             (downwards)scroll of background  contains a background horizontal counter offset
  5c00-5fff  1Kb Work RAM
  6000-63ff  1Kb Sound Control A (mirrored)             Write, sound control effect2 and 3
                           bits 0-3 : frequency divider
                           bits 4-5 : sound variation speed
                           bits 6-7 : noise generator
  6400-67ff  1Kb Work RAM
  6800-6bff  1Kb Sound Control B (mirrored)     Write, sound control effect1 and melody
                           bits 0-3 : frequency divider
                           bits 4-5 : trigger and filter
                           bit    6 : select melody, 1 = Jeux Interdits, 0 = La lettre ‡ Elise
                           bit    7 : start melody
  6c00-6fff  1Kb Work RAM
  7000-73ff  1Kb 8bit Game Control read-only (mirrored) Read, coin, start, player 1/2 controls
  7400-77ff  1Kb Work RAM
  7800-7bff  1Kb 8bit Dip Switch read-only (mirrored)
             bank switching with register at 0x7800 (data bit 0). Read, dip switch 0-6 and vblank
  7c00-7fff  1Kb Work RAM

------------------------------------------------
Hardware Registers :

  0x5000 (up to 0x53ff)  - Write, bit 0 = Player 1 or 2 RAM bank switching
                                bit 1 = Color palette high bit (6)

  0x5800 (up to 0x5bff)  - Write, background scroll offset

  0x6000 (up to 0x63ff)  - Write, sound control effect2 and 3

                             bits 0-3 : frequency divider
                             bits 4-5 : sound variation speed
                             bits 6-7 : noise generator

  0x6800 (up to 0x6bff)  - Write, sound control effect1 and melody

                             bits 0-3 : frequency divider
                             bits 4-5 : trigger and filter
                             bit    6 : select melody
                                        1 = Jeux Interdits
                                        0 = La lettre ‡ Elise
                             bit    7 : start melody

  0x7000 (up to 0x73ff)  - Read, coin, start, player 1/2 controls

  0x7800 (up to 0x7bff)  - Read, dip switch 0-6 and vblank


  memory mapped ports:
    read-only:
    7000-73ff IN
```

```
    7800-7bff Dip-Switch Settings (DSW)

    * IN (all bits are inverted)
    * bit 7 0x80 : barrier
    * bit 6 0x40: Left
    * bit 5 0x20: Right
    * bit 4 0x10: Fire
    * bit 3 0x08: -
    * bit 2 0x04: Start 2
    * bit 1 0x02: Start 1
    * bit 0 0x01: Coin

    * Dip-Switch Settings (DSW)
    * bit 7 0x80: VBlank
    * bit 6 0x40: free play (pleiads only)
    * bit 5 0x20: attract sound 0 = off 1 = on (pleiads only?)
    * bit 4 0x10: coins per play    0 = 1 coin  1 = 2 coins
    * bit 3 0x08:\ bonus
    * bit 2 0x04:/ 00 = 3000    01 = 4000  10 = 5000  11 = 6000
    * bit 1 0x02:\ number of lives (per coin?)
    * bit 0 0x01:/ 00 = 3  01 = 4  10 = 5  11 = 6

========================================================================
Phoenix Hardware Specification
Resolution 26x8 = (208 columnpix) x (32x8 = 256 rowlines)   (26 x-axis 32 y-axis chars per screen)
                                                    (D0.100, x=[0-BF],y=[0-FF])
    foreground screen (26x,32y) background +800 (starting at 4800)   O(0,0) = tl = 4320 tr(25,0) =
4000
    4320 4300 ... 4020 4000
    4321 4301 ... 4021 4001
    .... .... ... .... ....
    433e 431e ... 403e 401e
    433f 431f ... 403f 401f
------------------------------------------------------------------------
Charset A (foreground)
00-2f   (alpha)numeric chars       00-1A alphabet, 20-29 numeric
30-4f   spaceship (#8)             size (2x2)
50-5f   laser bullets (#16)        spaceship #8, enemy #8
60-bf   smallbird (#39)+copyright, 60-6f 1x1, 80-bf (2x2)
c0-df   explosions                 (1x1,4x2,3x2)
e0-ff   '' + shield
----------------------
Charset B (background)
00-1f   stars + small planets  (1x1)
20-3f   galaxies + nearby planets (2x2)
40-8f   boss-ship interior + cockpit + belt + hull
90-ff   bigbirds, egg + hatching + flapping-wings


========================================================================================================

Variables map (16 byte segments) 1st range
4340 - 434f   vars?
4350 - 435f   smallbird wave sequence related vars
4360 - 436f   vars (mainly hitcollision flags)
4370 - 437f   DEAD-ANIM slots, 4x
4380 - 438f   vars (Scores PL1/PL2/HI BCD format [aa.bb.cc.00]),  438c+d are soundControl vars
4390 - 439f   vars, 4390 PL1numlives, 4391 PL2numlives, ...
43a0 - 43af   vars, 43a0 Inputbackup, 43a1 copy of backup, 43a3 PLtoggle, 43a4 JT1-indexer, 43a5 JT1counter
43b0 - 43bf   vars
43c0 - 43cf   SPECIAL-OBJ (Spaceship + missiles + bombs)
43d0 - 43df     ''
43e0 - 43ef   COLLISION-OBJpos (zie onder)
43f0 - 43ff     ''
--------------------------------------------------------
Variables map (16 byte segments) 2nd range
4b40 - 4b4f   unused? vars
4b50 - 4b5f   @1000+offset pointers            // was: SOUND state  (per bird 2 bytes  2channel?)
4b60 - 4b6f     ''
4b70 - 4b7f   WINGDATA
4b80 - 4b8f     ''
```

```
4b90 - 4b9f     ''
4ba0 - 4baf     ''
4bb0 - 4bbf   WINGPOS (old/new) screenpnt enemy |  BIRBIRDs mode  other meaning
4bc0 - 4bcf     ''                              |
4bd0 - 4bdf     ''                              |  (4bd1, 4bd2, 4bd3, 4bd4, 4bd6, 4bd7)?
4be0 - 4bef     ''                              |  (4bed, 4bee [e0 35] + sp entry => PUSH-jumptable: 35e0 ?)
4bf0 - 4bff   SP Stack range

=====================================================
DETAILED VARs MAP
 ---------------------------------------------------------------------------
USAGE of VARs
     00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f
4340  .  .  .  .  .  .  .  .   .  .  .  .  .  .  .  .
4350  .  .  .  .  .  .  .  x   .  .  .  .  .  .  x  x
4360  x  +  +  x  x  .  x  +   ?  x  .  x  .  -  -  -


4380  x  x  x  *  x  x  x  *   x  x  x  *  x  x  ?  x
4390  x  x  .  x  x  .  .  x   x  x  x  x  .  .  .  .
43a0  x  x  x  x  x  x  x  x   .  .  x  x  .  .  .  x
43b0  .  .  .  .  x  .  .  .   x  x  x  .  x  .  x  .

-----
 4350 .  init value 04 ? (see 303E)
 4351 .  init value 2e ?    value of this var to (4b56/58/5a/5c), used at [Jump table 6]
 4352 .  init value 00 ?    value of this var to (4b5f)
 4353 .  init value 10 ?
 4354 .  init value 50 ?
 4355 .     counts down
 4356 .
v4357_wing.ZZ_hiboundary  (++?)
 4358 . countdown counter  (--?)
 4359 .
 435a .
 435b .
 435c ?
 435d bcd?
v435e flag (FF)
v435f_8bitscountup
-----
v4360_indication_LRbuttonPushingPushing? (FF = there is movement? or not 00?)
      4360 FF (pc=0916 movement to left, pc=0921 movement to right)
 4361 :v4361_laser-countdown
v4362_barrier_sound_timer
      4362 := 00,40--  soundcontrol B related var
      (before barrier shield at bigbirds first wave screen, pc=08de)
v4363_SpaceShip-hit := 00, (10 = just before spaceship hit), FF ,..
v4364_enemy-hit-detected := 00,(FF = enemy hit detected)
 4365 ?
v4366_BossShip-hit := 00, FF = detected
 4367 00,FF = at the start of the BossShip scrolling down (pc=22c4/d8)
 4368 00, (01 = bullet at horizontal height as lowest flying bigbird, repeat at anim change)
      02,04,08 as seen in code, has this has something to do with sounds?
v4369_BarrierShield-Hit := 00,FF (enemy bullets colliding with the barrier shield)
 436a ? 00/FF in code (pc=2796)  436a-- (pc=3B39)
v436b_8bitscounter
 436c ?
 436d - (pc=35a2  48,48,40,40,38,38,30,30,28,28,20,20,18,18,10,10) enemy wave?
 436e - (used in code)
 436f - (used in code)  [values 00,11,22,33,44,55,66,77,88,99,AA,BB,CC,DD,EE,FF]
-----

 4380/1/2/3 PL1 Score [aa.bb.cc.00]
 4384/5/6/7 PL2 Score  ''
 4388/9/a/b HIGH Score ''
            v4389 checksum? part of high score misused for this?
v438c_SoundControlA.       Copy of sound control 0x6000
v438d_SoundControlB.                          and 0x6800
 438e ?  (8 bit countdown value? )
v438f_numcredits Coin counter
```

```
-----
v4390_PL1numlives
v4391_PL2numlives
 4392 ?
v4393_JT6_indexer
v4394_4b50copy
v4397_busyflag (FF)
v4398_16bits_counter
 v4399_16bits_counter
v439a_16bits_counter_hi
 v439b_16bits_counter_low
 439c ?   seems to be used by square '*' rectangle window anim
 439d ?
 439e v439f_wing.YY_hiboundary
 439f v439f_wing.YY_lowboundary
-----
v43a0_INput_backup
v43a1_previous_INput
v43a2_curplayer. (totalplayers?)
v43a3_PLtoggle. (playernum)
v43a4_JT1-indexer Something to do with game phase?
v43a5_JT1counter, used for example when blinking players hi-score
v43a6_barrier_repeat_timer  --> Barrier Repeat Timer
v43a7_romscreendata-index8
v43aa_8bits-counter
v43ab_scrolltrigger
v43af_scrolltrigger
-----
v43b0_16bit_romptr_hi ??           pc=-5e2: copy-rom-ram()
         16bit_ptr (hival)      3F=start-level1+intro-phoenixtextscreen,2F=gameover-lvl1,
         points to ROM-lookup 30=start-bosslevel,2F=boss-scrolled +birds-appeared
v43b1_16bit_romptr_low      (loval) = 00, pc=2051(01/02) phoenixtext-introscreen + startlevel
v43b2 starfield_bossship_ptr_hi   (1c,1d)  -> 1c00 = starfield chars
v43b3 starfield_bossship_ptr_low  (00)     -> 1d00 = boss-ship chars
v43b4_JT4countdown
 43b5 flag  pc=5e2(FF) copy-rom-ram()
 43b6 flag  pc=-5e2:FF copy-rom-ram()
 43b7  not used?
 v43b8_staging0-F_videobit1 (00,01,04,08,09,0a,0b ...)  (bits #3-0) bit#1 = videopalette selection
v43b9_videoscroll
v43ba_smallbird_fleetsize
v43bb_remaining_bigbird_count
v43bc_?                         pc=22df(3F)
 43bd  not used?
v43be_bonus-lives
 43bf  not used?

--------------------------------------------------------------------------------
 SCORING
 4380 - 438b   vars (Scores PL1/PL2/HI BCD format [aa.bb.cc.00]? or [00.aa.bb.cc]?)
              4380 4381 4382 4383 PL1
             4384 4385 4386 4387 PL2
             4388 4389 438a 438b HI
 POINTS, when hitting/destroying:
 -small birds                    20 40 60 points ($14 $28 $3c)
 -flying sideways small birds   200 points  ($c8)
 -big birds                         50 100  points, bonus? [100-800]   ($42 $64)
 -big boss                      1000 - 9000 points

--------------------------------------------------------------------------------
SPEC.OBJ  43c0-43df  format [A=state B=index@1400 C=Xpixelpos  D=Ypixelpos]
                     A(others:  #w00x.yz00  w#7=alive,x#4=hidden,y#3=hit?,z#2=shield)
                     A(spaceship:) see Jump tables 2 en 3
                     B= index to 2x2 byte sprite format from base $1400
                     C= horizontal pixelpos between 00 (left) and C0 (right)
                     D= vertical pixelpos ( in steps of 8 pixels) E8=bottom 20?=TOP?
                        topleft(0,0) bottomright(C0,E8) ?
   #0 43c0/c1/c2/c3 = spaceship itself, example [9c 00 7c d8]
   #1 43c4/c5/c6/c7 = laser object spaceship (normal speed only 1 shot)
   #2 43c8/c9/ca/cb = rapid fire extra shot  (max 2 shots per screen)
```

```
   #3 43cc/cd/ce/cf = enemy birds bomb #1
   #4 43d0/d1/d2/d3 =  ''              #2
   #5 43d4/d5/d6/d7 =  ''              #3
   #6 43d8/d9/da/db = ? (not being addressed during demomode) -> big bird bomb #1
   #7 43dc/dd/de/df = ? ''                                    -> big bird bomb #2

   - A state   normal 8c 1000.1100  hidden 9c 1001.1100  shield 84 1000.0100  ??
   - 2x2 bytes sprite format @1400

externe inf0 -->
 43C0:  Barrier Status:  ANI 08h  Is barrier already ON?
 43C2:  Ship X Position

-------------------------------------------------------------------------------
COLLISION.OBJ
 43e0/e1  = SpaceShip screenposition (16bits)
 43e2/e3  = ''
 43e4/e5  = SpaceShip laser projectile (bottom) visible
 43e6/e7  = SpaceShip laser projectile (top) ahead
 43e8/e9  = one line above SpaceShip (leftpos)
 43ea/eb  = ''
 43ec/ed  = leftpos above (last line enemy birds)
 43ee/ef  =
 -
 43f0/f1  =  bird bomb pos1
 43f2/f3  =  ''
 43f4/f5  =  bird bomb pos2
 43f6/f7  =  ''
 43f8/f9  = ?
 43fa/fb  = ?
 43fc/fd  = ?
 43fe/ff  = ?

=====================================================
WINGDATA 4b70--4baf  Enemy wave objects  format [WW XX Yy Zz] (X and Y here at not coordinates!)
           W= JT2/3 jump state,
              bit#765 JT2 index, bit#4 JT2 HIDE indication, bit#3 JT3 SHOW indication, bit#210 JT3 index

           X=sprite_offset_0x1400, => anim offset
           Y=screenpos_column_offset_0x0a00, (bits #76543)  col[0-25]
           Z=screenpos-row_offset_0x0a00   (bits #76543)    row[0-31]
           O(0,0) at top left, gives for example-YyZz [70 34] col_X-cord 70= 01110.xxx > 14,
                                                              row_Y-cord 34= 00110.xxx = 6 ==> bird@(14,6)
           Y bits #210 meaning ?
           Z bits #210 meaning ?

0x0a00: pointers fg-screenpos-columns (26) from upperleft (4320) until topright (4000)
0x1400: sequences for multiplye bytes comprising the (enemy) object (sprite offset)


0a00: table for mapping from col index and row offset to video ram position of wingpos
    foreground column (@row0) screenpositions (4320 upperleft, 4000 is upperright)
 [Yy]index hi-bits(#7-#3) as index to determine column
 mapping from col to video ram position  col = 0 gives 0a00 + 0 = 0a00 points to 4320
               0    1    2    3    4    5    6    7     ... col
    0A00:      4320 4300 42E0 42C0 42A0 4280 4260 4240  col(0) - col(7)
    0A10:      4220 4200 41E0 41C0 41A0 4180 4160 4140  col(8) - col(15)
    0A20:      4120 4100 40E0 40C0 40A0 4080 4060 4040  col(16) - col(23)
    0A30:      4020 4000                                col(24) - col(25)
 [Zz] offset to determine row
 newpos will be *((0a00 + hi-bits YY(#7-#3) *2) + hi-bits ZZ(#7-#3) )

    Yy value for column entry (26 columns) 5bit value x-index
    Zz value for row    entry (32 rows)    5bit value y-offset

    example:
        Yy = 50-> 0.1010(000) -> entry #0a -> column 10 of screen (starting at )41e0
        Zz=  20-> 0.0100(000)->                   row 4 of screen     4
     YZ 5020 -> pos xy(10,4) points to 41e0 + 4  = 41e4 screenmem

Sprite sequence to be animated XX@1400
```

```
--------------------
 max 16 enemy units:
    4b70/71/72/73 #0
    4b74/75/76/77 #1
    4b78/79/7a/7b #2
    4b7c/7d/7e/7f #3

    4b80/81/82/83 #4
    4b84/85/86/87 #5
    4b88/89/8a/8b #6
    4b8c/8d/8e/8f #7

    4b90/91/92/93 #8
    4b94/95/96/97 #9
    4b98/99/9a/9b #a
    4b9c/9d/9e/9f #b

    4ba0/a1/a2/a3 #c
    4ba4/a5/a6/a7 #d
    4ba8/a9/aa/ab #e
    4bac/ad/ae/af #f

state:     bits xxxx   bit3 heeft speciale betekenis-> ?? hit by laser?
    sprite_offset              bits xxxx0000
    screenpos-highbyte_offset  bits xxxx0000
    screenpos-lowbyte_offset   bits xxxx0000

-----------------------------------------------------------------

WINGPOS (actual/new) screenpointers   (16 objects)       actual == [00 00]  only smallbirds?
 4bb0 - 4bbf   #0[aHaL nHnL]   #1[aHaL nHnL]  #2[aHaL nHnL]  #3[aHaL nHnL]  smallbirds #0-3
 4bc0 - 4bcf   #4[aHaL nHnL]   #5[aHaL nHnL]  #6[aHaL nHnL]  #7[aHaL nHnL]             #4-7
 4bd0 - 4bdf   #8[aHaL nHnL]   #9[aHaL nHnL] #10[aHaL nHnL] #11[aHaL nHnL]             #8-11
 4be0 - 4bef  #12[aHaL nHnL]  #13[aHaL nHnL] #14[aHaL nHnL] #15[aHaL nHnL]             #12-15
    #0[4bb0 4bb1 4bb2 4bb3] #1[4bb4 4bb5 4bb6 4bb7] #2[4bb8 4bb9 4bba 4bbb] #3[4bbc 4bbd 4bbe 4bbf]
    #4[4bc0 4bc1 4bc2 4bc3] #5[4bc4 4bc5 4bc6 4bc7] #6[4bc8 4bc9 4bca 4bcb] #7[4bcc 4bcd 4bce 4bcf]
    #8[4bd0 4bd1 4bd2 4bd3] #9[4bd4 4bd5 4bd6 4bd7] #a[4bd8 4bd9 4bda 4bdb] #b[4bbd 4bdd 4bde 4bdf]
    #c[4be0 4be1 4be2 4be3] #d[4be4 4be5 4be6 4be7] #e[4be8 4be9 4bea 4beb] #f[4bed 4bed 4bee 4bef]

-------------------------------------------------------------------------------
PC location accessing mem.  (mame:trackmem/pcatmemp)

Variables map (16 byte segments) 1st range
4340 - 434f   unused? vars,
              ?nentpos: equ 0x4340,
              ?ypos: equ 0x4342,
              ?delaystore: equ 0x4343
              ?scroll: equ 0x4345
              ?fireleftcount: equ 0x4346
              ?democounter: equ 0x4347  ;counts which round of demo mode we're in (for sound)
4350 - 435f   smallbird wave sequence related vars
4360 - 436f   vars (mainly hitcollision flags)
4370 - 437f   DEAD-ANIM slots, 4x
4380 - 438f   vars (Scores PL1/PL2/HI BCD format [aa.bb.cc.00])
4390 - 439f   vars  (boundaries?)
43a0 - 43af   vars.
              ?totalplayers: equ 0x43a2,
              ?playernum: equ 0x43a3 -> check player number - 0=P1, 1=P2
43b0 - 43bf   vars
43c0 - 43cf   SPECIAL-OBJ (Spaceship + missiles + bombs)
43d0 - 43df     ''
43e0 - 43ef   COLLISION-OBJpos
43f0 - 43ff     ''
    43e0+43e1 is pointer to current topleft point of spaceship in videomem
    43e2+43e3 is pointer to last? topleft point of spaceship in videomem, initially?
    43e4+43e5 points to outer left edge of screen 1 line above spaceship in videomem
    43e6+43e7 points to outer left edge of screen 1 line above spaceship in videomem, initially?
    43e8+43e9 points to outer left edge of screen 1 line above spaceship in videomem, initially?
    43ea+43eb points to outer left edge of screen 1 line above spaceship in videomem, initially?
    43ec+43ed points to outer left edge of screen 1 line beneath HUD in videomem
    43ee+43ef points to outer left edge of screen 1 line beneath HUD in videomem, initially?
```

```
    43f0 ...  ''

        00   01   02   03   04   05   06   07   08   09   0a   0b   0c   0d   0e   0f
4340     -    -    -    -    -    -    -    -    -    -    -    -    -    -    -    -
4350  3275 320a 320c 314d 31a4 30d2 3268    . 303c 30de 30de    .    .    .    . 2010
      05d9 05d9
4360   91e  95e    .    .    .    . 2381    .    .    .    .    .    .    .    .    .
4370     .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
4380     -    -    -    -    -    -    -    -    - 1ef8    -    -  37a  37c    -    -
4390     -    -    . 3004  511 326e    .    .  203  200    .  200    .    .  988  98e
43a0   3d8   97  b87  43b  51b  4af  8ea 2325  20c  208 24d4    +    +    +    +    +
43b0     +    +    +  6a9    +    +    +    -  850  67e  3fa  3fc 22db    .  16a    .
43c0   74f  92d  91b    +  728  92d  94c  96a    +    +    +    +  728  c8e 25f8  c94
43d0   74f  c8e 25f8  c94  74f  c8e 25f8  c94    +    +    +    +    +    +    +    +
43e0   891  88f  9c5  9d0  891  88f  9c5  9d0  891  88f  9c5  9d0  891  88f  9c5  9d0
43f0   891  88f  9c5  9d0  891  88f  9c5  9d0  891  88f  9c5  9d0  891  88f  9c5  9d0

 . = CLR = Clear memory routine @5d9
 + = CPY = Copy memory @5e1

Variables map (16 byte segments) 2nd range
4b40 - 4b4f   unused? vars
4b50 - 4b5f   SOUND state (per bird 2 bytes)
4b60 - 4b6f     ''  sound related?

4b70 - 4b7f   WINGDATA
4b80 - 4b8f     ''
4b90 - 4b9f     ''
4ba0 - 4baf     ''

4bb0 - 4bbf   WINGPOS (actual/new) screenpointers
4bc0 - 4bcf     ''
4bd0 - 4bdf     ''
4be0 - 4bef     ''

4bf0 - 4bff   SP Stack range


        00   01   02   03   04   05   06   07   08   09   0a   0b   0c   0d   0e   0f
4b40   69a 3516    -    - 3513 3516    -    - 3513 3516 3513 3516 3513 3516    -    -
4b50   667  d66  667  d66  667  d66  667  d66  667  d66  667  d66  667  d66 329e  d59
4b60   667  d66  667  d66 329e 32a3  667  d66  667  d66  667  d66  667  d66  667  d66
4b70   74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f
4b80   74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d4e  d53
4b90   74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f
4ba0   74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f  74f  dd7  d61  62f
4bb0   a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0
4bc0   a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0
4bd0   a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0
4be0   a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0  a81  a7f  9c5  9d0


--------------------------------------------------------------------------------

"


Let's rock!!!
0000================================================================================================
0000-0007:         nop
                                    #==== interrupt RST 1 entry ==== GAMELOOP
0008:     lxi  sp,$4bff          #   stack pointer initialisatie
000B:     mvi  h,$50             Video Control (palette)
000D:     mvi  m,$00             #   palette 0
000F:           call $0050       #   reset video&sound registers?
0012:     lxi  h,$1800           #   (1800: ptr 4320) linker bovenhoek scherm
0015:     mvi  c,$03             #   drie regels tekst (vanaf HL+6)
0017:           call $01d0       #   PrintLNs aanroepen

    1800:  4320 FFFFFFFF 00 13 03 0F 12 05 21 00 00 08           "SCORE1  H"
    1810:  09 2B 13 03 0F 12 05 00 00 13 03 0F 12 05 22 00  "I-SCORE  SCORE2"
```

```
    1820:  4321

(002A, 0043, 0049)
001a----------------------------------------------------main game loop
001A:           call $0080          # VBLANK afhandeling
001D:    ldax $43a2                 # v43a2_curplayer
                                #-- interrupt rst 4 entry ---
0020:    ana  a                     # (Z when a=0, otherwise NZ)
0021:    jz   $002d                 # geen player actief, bvb bij na game-over

                                       #--- TRAP interrupt entrys ---
# PL1 of PL2 nog actief afhandeling
0024:           call $0400          # Jump Table 1

JT5: (<-0EE5) #### when SHIP HIT (not game-over)
0027:           call $2700          # PL1/PL2 spelen
002A:    jmp  $001a             main-game-loop

(0021)
002d----------------------------------------------------DEMO MODE (curplayer=0)
002D:    mvi  a,$0f
002F:    mvi  h,$60                 Sound Control A reg
0031:    mov  m,a               #sound
0032:    mvi  h,$68                 Sound Control B reg
                                #--- RST 6.5 interrupt entry ---
0034:    mov  m,a               #sound
0035:           call $0377          # set soundcontrolvars to 0F

                                #=== interrupt RST 7 entry ===
0038:    nop
0039:    ldax $438f                 # v438f_numcredits  (99=inifinite?)
                                #--- RST 7.5 interrupt entry ---
003C:    ana  a                     # (Z when a=0, otherwise NZ)
003D:    jz   $0046                 # no credits left
                                    # otherwise
0040:           call $0288          # (Ask&play) one/two player input handling
0043:    jmp  $001a             main-game-loop

0046----------------------------------------------------
0046:           call $00e3          continue demo mode?, main attract mode

JT5: (<-0EE5) #### when SHIP HIT (and game-over)
0049:    jmp  $001a             main-game-loop



000F
0050---------------------------------------------------- reset video&sound registers
0050:    mvi  h,$68             Sound Control B
0052:    mvi  m,$00                 # 6800 := 00
0054:    mvi  h,$60             Sound Control A
0056:    mvi  m,$00                 # 6000 := 00
0058:    mvi  h,$58             Video Scroll Register
005A:    mvi  m,$00                 # 5800 := 00
005C:           call $006b          # wis hoger variabelenbereik
005F:    mvi  h,$50             Video Control (palette)
0061:    mvi  m,$01                 # palette 1
0063:           call $006b          # wis hoger variabelenbereik
0066:    mvi  h,$50             Video Control (palette)
0068:    mvi  m,$00                 # palette 0
006A:    ret



005c, 0063
006b----------------------------------------------------clear (00) variables $4bf8 to -=3f ($4BB9)
006B:    lxi  h,$4bf8               # SP als einde
006E:    mvi  a,$3f                 # count
(0074)
0070:    mvi  m,$00
0072:    dcx  h
```

```
0073:   cmp  h
0074:   jnz  $0070
0077:   ret



011f
0078----------------------------------------------------
0078:       call $0196          # afdrukken PHOENIX tekst
007B:    jmp  $06f0                 # scroll background and plot


001A, 0143, 016F
0080-------------------VBLANK-- interrupt--Wait for VBlank-----------------------------------
(0085)
0080:    mvi  h,$78         DSW    DIPs/Vblank signal
0082:    mov  a,m
0083:    ani  $80               # bit 7 DSW =  VBLANK
0085:    jz   $0080         # geen VBLANK wachtloop
(008b)
0088:    mov  a,m
0089:    ani  $80
008B:    jnz  $0088         wel VBLANK
008E:    mvi  h,$70             INput (8bit Game Control read-only)
0090:    mov  a,m
0091:    lxi  h,$43a0           # v43a0_INput_backup
0094:    mov  b,m
0095:    mov  m,a               # set v43a0_INput_backup
0096:    inr  l
0097:    mov  m,b               # v43a1_previous_INput (gebruikt bij 00c1)
0098:    mvi  l,$9b             # v439ab_16bits_counter
009A:           call $0200      # telt aantal keren aanroep in 16 bits teller (439a:439b)
009D:    mvi  b,$01             # bit 0 'coin' bit
009F:           call $00bb      # INput var bit [B]  waarde veranderd?
00A2:    rz                     # nee, terugkeren

00A3:    mvi  l,$8f             # v438f_numcredits
00A5:    mov  a,m
00A6:    cpi  $99               # infinite credits bij waarde 99?
00A8:    rz                     # ja, terugkeren

00A9:    lxi  b,$0001
00AC:           call $0220      # BCD conversie? ([B] erbij op tellen?)
00AF:    lxi  d,$4142           # COIN xN waarde (4142 is schermpositie COINxx rechts)
00B2:    mvi  b,$02
00B4:           call $00c4      # plotten  (b=2 #characters)
00B7:    ret



009f, 08d6, 0939
00bb-------------------------- Check the INput_backup var [B] bit value has been tiggered
#009d B has set to bit #0  -> coin input
#08d4 B has set to bit #7  -> Barrier button
#0937 B has set to bit #4  -> FIRE button
00BB:    lxi  h,$43a0           # v43a0_INput_backup
00BE:    mov  a,m
00BF:    cma                    # bepaal compliment (vanwege INput inverted waarde)
00C0:    ana  b                 # (#01 bij aanroepende 009f is bit voor COIN)
00C1:    inr  l                 # v43a1_previous_INput (zie 0097)
00C2:    ana  m                 # 'actuele'  waarde met voorgaande waarde vergelijken
00C3:    ret



00b4, 02e9, 030d, 0341, 034b, 04e1, 2778
00c4---------------------------------------- value conversion loop [B] plot charset chars
#  plot(@d,#b)
```

```
00C4:    mov  a,m              # bevat #coins  vb $49
00C5:    ani  $0f              # low-end nibble  waarde eerst plotten
00C7:    ori  $20              # 0-9 beginnen bij charset positie 20 hex
00C9:    stax d                # charset waarde $2N plotten op scherm (bij COINnn)
00CA:          call $0210      # cursor met 1 kolom positie naar links
00CD:    dcr  b                # B--
00CE:    rz

00CF:    mov  a,m              # a := #coins
00D0:    rrc                   # 4xnaar rechts roteren voor hi-end nibble waarde, vb 49 -> 4
00D1:    rrc
00D2:    rrc
00D3:    rrc
00D4:    ani  $0f              # idem
00D6:    ori  $20              # waarde naar charset waarde voor 0-9
00D8:    stax d                # plotten waarde $2N voor COINnn
00D9:          call $0210      # cursor met 1 kolom positie naar links
00DC:    dcx  h                # HL-- voor VAR met pos-1 aanduiding
00DD:    dcr  b                # B-- loop waarde mbt DO UNTIL
00DE:    jnz  $00c4
00E1:    ret                   # B=0




0046
00e3----------------------------------------------------continue demo mode?
#     per jump (jz/jnc/..) in stukje code met ret (terug naar 0049 jmp 001A)
00E3:    lxi  h,$4399              # v43989_16bits_counter
00E6:          call $0200      # telt aantal keren aanroep in 16 bits teller (4398:4399)
00E9:    lxi  b,$0001              # c==01
00EC:          call $0258          # vergelijk BC met actuele 16 bits teller
00EF:    jz   $01e1            # init steps (nieuw scherm + vars op default)

00F2:    lxi  b,$0002              # BC=0002
00F5:    lxi  d,$011f              # DE=011f
00F8:          call $0260          # todo_BCDE_teller_handling
00FB:    jnc  $0196            # afdrukken PLAYER/COIN/SCORE AVERAGE informatie

00FE:    lxi  b,$0120              # BC=0120
0101:          call $0258          # vergelijk BC met actuele 16 bits teller
0104:    jz   $0bca            # Enemies in score average table

0107:    mvi  c,$b0                # BC=01b0
0109:          call $0258          # vergelijk BC met actuele 16 bits teller
010C:    jz   $01e1            # init steps (nieuw scherm + vars op default)

010F:    mvi  c,$b8                # BC=01b8
0111:          call $0258          # vergelijk BC met actuele 16 bits teller
0114:    jz   $0580            # init vars [43ab .. 43b6]

0117:    mvi  c,$c0                # BC=01c0?
0119:    lxi  d,$02df              # DE=02df
011C:          call $0260          # todo_BCDE_teller_handling
011F:    jnc  $0078            # combi: plot PHOENIX text, scroll background + plot

0122:    lxi  b,$0300              # BC=0300
0125:    lxi  d,$03af              # DE=03af
0128:          call $0260          # todo_BCDE_teller_handling
012B:    jnc  $21dc            # hatched bigbird (fade-in/out) animation under PHOENIX text

012E:    lxi  b,$03e6              # BC=03e6
0131:    lxi  d,$ffff              # DE=ffff
0134:          call $0260          # todo_BCDE_teller_handling
0137:    jnc  $03b0            # stay in EXAMPLE LEVELs
013A:    ret




01e1, 0288, 02b9, 02c0
```

```
0140----------------------------------------------------- zet schermen en bepaalde vars op default
0140:           call $03a0      # schoon background met 00
0143:           call $0080      # vblank
0146:           call $0380      # schoon foreground behalve HUD (1e 3 regels)
0149:    lxi  h,$43a3           # v43a3_PLtoggle
014C:    mvi  m,$02             #               := 0000.0010
014E:    inr  l                 # v43a4_JT1-indexer
014F:    mvi  m,$00             #               := 00
0151:    nop
0152:    nop
0153:    nop
0154:    mvi  l,$b8             # 43b8
0156:    mvi  b,$08             # count =8
0158:           call $05d8      # dus CLEAR memory [43b8..43c0]
015B:    mvi  l,$ba             # v43ba_smallbird_fleetsize
015D:    mvi  m,$10             #   := 10
015F:    mvi  l,$be             # v43be_bonus-lives
0161:    ldax $7800             DSW
0164:    ani  $0c               #       0000.1100 bit #3,2 -> Bonus_Life 6K 60K
0166:    rlc                    #        000x.x000
0167:    rlc                    #        00xx.0000  (max 30)
0168:    adi  $30
016A:    mov  m,a               #v43be_bonus-lives [30 .. 60] := 30 (default)
016B:    mvi  h,$58             Video Scroll Register
016D:    mvi  m,$00             #               := 00
016F:           call $0080      # vblank
0172:    ret




03ce
-----------------------------------------------------set [B] obv bereik v43989_16bits_counter
"  lowval            [B]   voor OR met v43a0_INput_backup (FAKEing input)
  [00..1E, 60..7E]   CE               1100.1110-> 0011.0001 right+fire   +coin
  1F,5F,7F           FE               1111.1110-> 0000.0001               coin
  [20..5F>           AE               1010.1110-> 0101.0001 left+fire    +coin
  7F + highval 09    7E               0111.1110-> 1000.0001 barrier      +coin
"
0173:    mov  a,m               # low value v43989_16bits_counter #00e3
0174:    ani  $7f               # afkappen tot max 7F
0176:    mvi  b,$ce
0178:    cpi  $1f
017A:    rc                     # [0 <= X < 1F]

017B:    mvi  b,$fe
017D:    rz                     # 1F

017E:    mvi  b,$ae
0180:    cpi  $5f
0182:    rc                     # [20 <= X < 5F]

0183:    mvi  b,$fe
0185:    rz                     # 5F

0186:    mvi  b,$ce
0188:    cpi  $7f
018A:    rc                     # [60 <= X < 7F]

018B:    mvi  b,$fe             # 7F
018D:    dcr  l                 # hi value v43989_16bits_counter
018E:    mov  a,m
018F:    cpi  $09               # counter geen 09.xx
0191:    rnz

0192:    mvi  b,$7e             # counter is 09.xx
0194:    ret
```

```
0078 (00fb)
0196---------------------------------------------------- slowmotion plot van PLAYER/COIN/AVERAGE SCORE
0196:    mov  a,m              # een van de tellers als ingang 4399/9b?
0197:    ani  $1f              # 0001.1111
0199:    cpi  $06
019B:    rc

# (v43989_16bits_counter lowval) na AND tussen [06-1F]
019C:    mov  e,a              # vb 06
019D:    mov  a,m              # 4399 vb 26
019E:    ani  $e0              # 1110.0000  stapjes van 20 [20/40/60/80/a0/c0/eo]
01A0:    mov  c,a
01A1:    dcr  l                # point to 4398
01A2:    mov  b,m
01A3:    mvi  l,$a8            # make backup of
01A5:    mov  m,b              #   current BC content
01A6:    inr  l                #   to (43a8:43a9)
01A7:    mov  m,c              #   vb (00:20) (00:40) (00:60) ...
01A8:    lxi  b,$1860          #   BC:=1860 bij (43a8:43a9)
01AB:           call $0206     #   tel BC op bij 16bits geheugen
01AE:    mov  a,m
01AF:    dcr  l
01B0:    mov  h,m
01B1:    mov  l,a
01B2:    mov  a,e
01B3:    mov  d,m              # (HL=1860: 43 25) (HL=1880: 4327) ... * 1PLAYER 1COIN..
01B4:    inr  l
01B5:    mov  e,m
01B6:    dcr  l
01B7:    mov  c,a              # DE := *(1860/80/a0/...) vb 4325
01B8:    add  l
01B9:    mov  l,a
01BA:    mov  a,c
01BB:    sui  $06
01BD:    mov  c,a
01BE:    jz   $01c8
(01c5)
01C1:           call $0217     # cursor met 1 kolom positie naar rechts
01C4:    dcr  c
01C5:    jnz  $01c1
(01be)
01C8:    mov  a,m              # afdrukken tekst
01C9:    stax d                #vb *  1PLAYER  1COIN    *  2PLAYERS .....
01CA:    ret

    1860:  (4325) FFFFFFFF 00 00 00 00 00 00 00 09 0E 13     "              INS"
    1870:  05 12 14 00 00 03 0F 09 0E 00 00 00 00 00 00 00   "ERT  COIN        "
    1880:  (4327) FFFFFFFF 00 00 00 1F 00 21 10 0C 01 19     "          * 1PLAY"
    1890:  05 12 00 00 00 21 03 0F 09 0E 00 00 1F 00 00 00   "ER   1COIN  *    "
    18A0:  (4329) FFFFFFFF 00 00 00 1F 00 22 10 0C 01 19     "          * 2PLAY"
    18B0:  05 12 13 00 00 22 03 0F 09 0E 13 00 1F 00 00 00  "ERS  2COINS *   "
    18C0:  (432E) FFFFFFFF 00 00 00 13 03 0F 12 05 00 01    "   SCORE A"
    18D0:  16 05 12 01 07 05 00 14 01 02 0C 05 00 00 00 00    "VERAGE TABLE"

    # 01CB: 01 0D C2   lxi  b,$c20d
    # 01CE: C0         rnz
    # 01CF: 01 56 2C   lxi  b,$2c56



0017(hl=1800,c=03), 0290, 02a2, 0b95   print Cx number of textlines
01d0----  PrintLNs function --------------------------------
(01DD)
01dO:   mov  d,m              # d:= *(src)   vb 1800: 43 20
01d1:   inr  l
01D2:    mov  e,m             # e:= *(src+1)     DE := 4320 scherm links boven
01D3:    mov  a,l
01D4:    adi  $05             # src += 5       de tekst begint bij positie HL+6
01D6:    mov  l,a
```

```
01D7:    mvi  b,$1a            # doel DE=4320,  B=26 (breedte van scherm per regel)
01D9:           call $01ed     # PRINT+CURSOR
01DC:    dcr  c
01DD:    jnz  $01d0
01E0:    ret




00ef, 010c
01e1-------------------------------------------------
01E1:           call $0140     # zet schermen en bepaalde vars op default

0b98
01e4------------------------------HUD2 bottom print: (c) 1980 TAITO CORPORATION
01E4:    lxi  h,$1960          # romptr 1960: 43 3C   rechtsonder twee na laatste rij
01E7:    mvi  c,$03
01E9:    jmp  $01d0            # PrintLNs


    1960:  (433C) 00000000 00 00 00 00 00 00 00 00 00 00
    1970:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    1980:  (433D) 21002100 00 2C 00 21 29 28 20 00 14 01         " (c) 1980 TA"
    1990:  09 14 0F 00 03 0F 12 10 0F 12 01 14 09 0F 0E 00    "ITO CORPORATION"


01d9
01ed-------------------------------------------------------PRINT+CURSOR
(01F4)
01ED:    mov  a,m                  # HL wijst naar tekst
01EE:    stax d                    # plot op scherm naar cursor [DE] positie
01EF:    inx  h                    # pointer+1 over te plotten tekst
01F0:           call $0217         # cursor naar rechts door -$20 te doen
01F3:    dcr  b
01F4:    jnz  $01ed
01F7:    ret




009a, 00e6
-------------------------------------------------------teller 0000->00FF->0100->....FFFF

"  16 bits counters (4398:4399) demo-counter: wordt verhoogd in 00e3 subroutine
  en (439a:439b) VBLANK(timer) counter: wordt verhoogd in 0080 blok
  telt aantal keren aanroep in 16 bits teller, eentje vanuit vblank en eentje vanuit demo mode
"
0200:    inr  m                # *HL++                          (4399)
0201:    rnz                   # pas als de teller 'rond' is

0202:    dcr  l                # HL--                           (4398)
0203:    inr  m                # *HL++
0204:    inr  l
0205:    ret



01ab
-------------------------------------------------------tel BC op bij 16bits geheugen
 (*HL-1)+B, (*HL)+C                                              (43a8:43a9)
0206:    mov  a,m
0207:    add  c
0208:    mov  m,a   #memcopy
0209:    dcr  l
020A:    mov  a,m
020B:    adc  b
020C:    mov  m,a   #memcopy
020D:    inr  l
020E:    ret
```

```
00ca, 00d9, 04fe, 0aa5, 0af9, 0fe3, 37c1
---------------------------------------------- cursor met 1 kolom positie naar links verplaatsen
0210:    mov  a,e
0211:    adi  $20
0213:    mov  e,a
0214:    rnc

0215:    inr  d
0216:    ret



01c1, 01f0, 0780, 0795, 07c7, 07e4, 0ae2, 0f6a, 0f79, 1ee9, 37bb
---------------------------------------------- cursor met 1 kolom positie naar rechts verplaatsen
0217:    mov  a,e
0218:    sui  $20
021A:    mov  e,a
021B:    rnc

021C:    dcr  d
021D:    ret



00ac, 272d, 275c
------------------------------------------------ conversie high? HHMMLL formaat? score naar BCD?
0220:    xra  a              # eor
0221:    mov  a,m            # HL
0222:    add  c
0223:    daa                 # decimal adjust
0224:    mov  m,a            # HHxxxx score
0225:    dcr  l
0226:    mov  a,m            # HL-1
0227:    adc  b              # + xxMMxx score    +[B]
0228:    daa                 # decinal adjust
0229:    mov  m,a            #score
022A:    dcr  l
022B:    mov  a,m            # HL-2
022C:    aci  $00            # xxxxLL score waarbij LL altijd 00 is (score minimaal 000100)
022E:    daa                 # decimal adjust
022F:    mov  m,a            #score
0230:    inr  l              # HL-1  herstel
0231:    inr  l              # HL        ''
0232:    ret



02DF
-----------------------------------------------------iets met credits/lives?
# (02df heeft als input v438f_numcredits, c=02)
0236:    stc                 # set carry flag
0237:    mvi  a,$99          # lives #99
0239:    aci  $00            # add with carry ($100)
023B:    sub  c              # -02
023C:    add  m              # v438f_numcredits
023D:    daa                 # decimal adjust calculator
023E:    mov  m,a
023F:    dcr  l              # 438e?
0240:    mvi  a,$99
0242:    aci  $00
0244:    sub  b
0245:    add  m
0246:    daa                 # decimal adjust calculator
0247:    mov  m,a
0248:    dcr  l              # 438d?
0249:    mvi  a,$99
024B:    aci  $00
```

```
024D:    add  m
024E:    daa               # decimal adjust calculator
024F:    mov  m,a
0250:    inr  l
0251:    inr  l            # terug naar 438f
0252:    ret



00ec, 0101, 0109, 0111, 03b9, 03c8
-----------------------------------------------------vergelijk BC met actuele 16 bits teller
# als [m] de waade [c] heeft dan controleren of [m-1] gelijk is aan [b]
0258:    mov  a,m       # a= *HL
0259:    cmp  c
025A:    rnz            # a <> c

025B:    dcr  l
025C:    mov  a,m       # a:= *(HL-1)
025D:    inr  l
025E:    cmp  b         # a=b?
025F:    ret



00f8, 011c, 0128, 0134
-----------------------------------------------------todo_BCDE_teller_handling
0260:       call $0270          # Carryflag obv (16bits teller < BC)
0263:    rc                         #
0264:       call $0277          # Carryflag obv (DE < 16bits teller)
0267:    ret



0260, 03b3, 03c2
-----------------------------------------------------Carryflag obv (16bits teller < BC)
0270:    mov  a,m
0271:    sub  c
0272:    dcr  l
0273:    mov  a,m
0274:    sbb  b
0275:    inr  l
0276:    ret



0277: 0264
-----------------------------------------------------Carryflag obv (DE < 16bits teller)
0277:    mov  a,e
0278:    sub  m
0279:    dcr  l
027A:    mov  a,d
027B:    sbb  m
027C:    inr  l
027D:    ret



???
-------------------------------------------------oude code om HL met BC te vergelijken?
    0280:    mov  a,l
    0281:    cmp  c
    0282:    rnz

    0283:    mov  a,h
    0284:    cmp  b
    0285:    ret



0040
-----------------------------------------------------Ask one/two player input handling
```

```
0288:           call $0140      # zet schermen en bepaalde vars op default
028B:    lxi  h,$19c0           #  (romptr 19C0:  43 28) regel #8 links vanaf boven (#0)
028E:    mvi  c,$02             #  2 regels tekst
0290:           call $01d0      # PrintLNs
0293:    mvi  c,$02
0295:    ldax $438f             # v438f_numcredits
0298:    cpi  $02               # 2 credits over betekent een 1 of 2 player game
029A:    jc   $02a7
029D:    lxi  h,$1ba0           #  (1BA0:  43 2C)  regel #C links vanaf boven (#0)
02A0:    mvi  c,$01             #  1 regel tekst
02A2:           call $01d0      # PrintLNs
02A5:    mvi  c,$06             # 0110  bit #2=start PL2 ,bit #1=start PL1
(029a)                          #
02A7:    ldax $7000             8bit Game Control
02AA:    cma                    # compliment A  waarde 0 betekende gezet, nu 1 via inverse
02AB:    ana  c
02AC:    rz                     #als er geen start van PL1 of PL2 is dan return
                                        #anders
----------------------------------------player 1 of player 2 start
02AD:           call $02cb      # print #coins
02B0:           call $02f0      #hiscore plot?
02B3:           call $032e
02B6:           call $0350
02B9:           call $0140      #   zet schermen en bepaalde vars op default
02BC:    mvi  h,$50             Video Control (palette)
02BE:    mvi  m,$01             # palette 1
02C0:           call $0140      #   zet schermen en bepaalde vars op default
02C3:    mvi  h,$50             Video Control (palette)
02C5:    mvi  m,$00             # palette 0
02C7:    ret



02ad
------------------------------------------------------set/print? v43a2_curplayer
02CB:    mvi  c,$01             # 01 maar deze waarde wordt gelijk overruled met 02
02CD:    cpi  $02
02CF:    jz   $02d4
02D2:    mvi  c,$02
(02cf)
02D4:    lxi  h,$43a2           # v43a2_curplayer
02D7:    mov  m,c               #                := 02      PL2
02D8:    inr  l
02D9:    mvi  m,$00             # v43a3_PLtoggle :=00
02DB:    mvi  b,$00
02DD:    mvi  l,$8f             # v438f_numcredits
02DF:           call $0236      # maak er BCD van?
02E2:    mvi  l,$8f             # v438f_numcredits
02E4:    lxi  d,$4142           # COIN veld op het scherm 4142 is positie voor N in COINxN
02E7:    mvi  b,$02
02E9:           call $00c4      # plot (nr.coin--> pos 4142)
02EC:    ret



02B0
------------------------------------------------------hiscore plot?
02F0:    lxi  d,$4383
02F3:    lxi  h,$438b               # last byte of High    score in BCD
02F6:           call $0314         # BCD calculation?
02F9:    cnc  $0320                 # copy 3 bytes van D naar HL
02FC:    mvi  e,$87
02FE:    mvi  l,$8b                 # 438b hiscore?
0300:           call $0314         # iets met BCD berekenen?
0303:    cnc  $0320                 # copy 3 bytes van D naar HL
0306:    mvi  l,$8b                 # 438b hiscore?
0308:    lxi  d,$4141               # HIGHSCORE 000000 rechter positie
030B:    mvi  b,$06                 # 6 karakters
030D:           call $00c4         # plotten
0310:    ret
```

```
02F6, 0300, 2786
-----------------------------------------------------iets met BCD berekenen?
0314:    ldax d
0315:    sub  m
0316:    dcr  e
0317:    dcr  l
0318:    ldax d
0319:    sbb  m
031A:    dcr  e
031B:    dcr  l
031C:    ldax d
031D:    sbb  m
031E:    ret



02f9, 0303
-----------------------------------------------------copy 3 bytes van D naar HL
0320:    ldax d
0321:    mov  m,a
0322:    inx  d
0323:    inx  h
0324:    ldax d
0325:    mov  m,a
0326:    inx  d
0327:    inx  h
0328:    ldax d
0329:    mov  m,a
032A:    ret



02B3
-----------------------------------------------------iets met plot  hiscore?
032E:    lxi  h,$4380           # PL1 Score BCD HI
0331:    mvi  m,$00
0333:    inx  h
0334:    mov  a,l
0335:    cpi  $88
0337:    jnz  $0331
033A:    mvi  l,$83             # ???PL1 Score BCD LO
033C:    lxi  d,$4261           # schermpos PL1score 000000 rechterkant
033F:    mvi  b,$06
0341:           call $00c4      # plot
0344:    mvi  l,$87             # 4387 PL2
0346:    lxi  d,$4021
0349:    mvi  b,$06
034B:           call $00c4      # plot
034E:    ret



02B6
-----------------------------------------------------DSW switch handling
0350:    ldax $7800          DSW
0353:    ani  $03            nr.lives per coin bit#1+#0 (00 = 3  01 = 4  10 = 5  11 = 6)
0355:    adi  $03            # waarde is inverted
0357:    mov  b,a
0358:    lxi  h,$4390        # v4390_PL1numlives    INFINITE LIVEs PL1 bij 09
035B:    mov  m,b
035C:    mvi  l,$a2          # v43a2_curplayer
035E:    mov  a,m
035F:    cpi  $01
0361:    jz   $0367          # werk nr. of lives bij

0364:    mvi  l,$91          # v4391_PL2numlives
```

```
0366:    mov  m,b

(0361),0b22, 2791
-------------------------------------------------------werk nr. of lives bij
0367:    mvi  l,$90         # v4390_PL1numlives
0369:    mov  a,m
036A:    ori  $20
036C:    stax $42a2         # schermlocatie #ships PL1
036F:    inr  l             # v4391_PL2numlives
0370:    mov  a,m
0371:    ori  $20
0373:    stax $4062         # schermlocatie #ships PL2
0376:    ret




0035
-------------------------------------------------------set soundcontrolvars to 0F
0377:    lxi  h,$438c       # v438c_SoundControlA
037A:    mov  m,a           #           := [A=0f in 002d]
037B:    inr  l             # v438d_SoundControlB
037C:    mov  m,a           #           := OF
037D:    ret



0380: 0146, 07f6, 2521
-------------------------------------------------------schoon foreground behalve HUD (1e 3 regels)
0380:    lxi  h,$433f   # VIDEO reg Charset A (links-onder)
0383:    lxi  d,$001f   # kolomlengte
0386:    lxi  b,$033f   # grootte voorgrond scherm
(0390,039a)
0389:    mov  m,d
038A:    dcx  h
038B:    mov  m,d
038C:    dcx  h
038D:    mov  a,l
038E:    ana  e
038F:    cmp  b             # kolom moet > 03 zijn
0390:    jnz  $0389

0393:    mov  m,d
0394:    dcx  h
0395:    dcx  h
0396:    dcx  h
0397:    dcx  h             # naar onderkant volgende kolom
(039a)
0398:    mov  a,h
0399:    cmp  c
039A:    jnz  $0389
039D:    ret




0140, 22f0
-------------------------------------------------------schoon background met 00 [4b3f - 4800]
03A0:    lxi  h,$4b3f       # VIDEO RAM Charset B (links-onder)
03A3:    lxi  d,$0047
03A6:    mov  m,d
03A7:    dcx  h
03A8:    mov  m,d
03A9:    dcx  h
03AA:    mov  a,h
03AB:    cmp  e             # (tot aan H=E bij 47ff)
03AC:    jnz  $03a6
03AF:    ret
```

```
0137
------------------------------------------------------EXAMPLE LEVELS
"  clear screen and blinking PL1 score (000000)
  scrolling through and faking input
  counter 07a0-> bigbirds, 0b60->smallbirds ??
    teller (4398:4399 v43989_16bits_counter)
"
03B0:    lxi  b,$07a0                  # BC=07a0 waarde voor teller vergelijk
03B3:           call $0270             # Carryflag obv (16bits teller < BC)
03B6:    jc   $03ce                 # de teller is nog niet bij 07ba0
03B9:           call $0258             # vergelijk BC met actuele 16 bits teller
03BC:    jz   $03eb                 # alleen bij 07a0 waarde set BIGBIRD mode

03BF:    lxi  b,$0b60                  # BC=0b60 waarde voor teller vergelijk
03C2:           call $0270             # Carryflag obv (16bits teller < BC)
03C5:    jc   $03ce                 # de teller is nog niet bij 0b60
03C8:           call $0258             # vergelijk BC met actuele 16 bits teller
03CB:    jz   $03e2                 # alleen bij 0b60 waarde set SMALLBIRD mode

(03B6,03C5) counter not reached questioned value or after bird mode set
03CE:           call $0173          # set [B] on v43989_16bits_counter to fake input
03D1:    lxi  h,$43a0               #   v43a0_INput_backup
03D4:    mov  a,m
03D5:    ani  $01                   #   bit #0 coin ongemoeid laten
03D7:    ora  b                     #   FAKEing input
03D8:    mov  m,a                   #   v43a0_INput_backup :=  01 OR [B]
------------------------------------------------------
03D9:    jmp  $0400                 Jump Table 1
    # 03DC:    jmp  $0400                      Jump Table 1


03cb
------------------------------------------------------SMALLBIRDS mode
03E2:    lxi  b,$0108           # BC := [b-> JT1_index            c->videopal]
03E5:    lxi  d,$1000           # DE := [d-> fleetsize smallbirds e->fleetsize bigbirds]
03E8:    jmp  $03f1             # goto set fleetsize

03bc
------------------------------------------------------BIGBIRD mode + ending smallbird
  bigbirds (8 pieces)
03EB:    lxi  b,$0104           # BC := [b-> JT1_index            c->videopal]
03EE:    lxi  d,$0008           # DE := [d-> fleetsize smallbirds e->fleetsize bigbirds]
  (03E8)                                # jumped means SMALLBIRD settings
03F1:    lxi  h,$43a4           # v43a4_JT1-indexer
03F4:    mov  m,b               #       43a4:= 01
03F5:    mvi  l,$b8             #  v43b8_staging0-F_videobit1  (set bit#3 or bit#2)
03F7:    mov  m,c               #   := [08 of 04 #3/#2]    (03e2->08, 03eb->04 )
03F8:    mvi  l,$ba             # v43ba_smallbird_fleetsize
03FA:    mov  m,d               #   := [00 of 10]          (03e5->10 of 03ee->00)
03FB:    inr  l                 # v43bb_remaining_bigbird_count
03FC:    mov  m,e               #   := [08 of 00]          (03e5->00 of 03ee->08)
03FD:    ret




0024,(03D9,03DC)
------------------------------------------------------JUMP TABLE 1
0400:    lxi  h,$040e       # base + offset wordt nieuwe PC counter
0403:    ldax $43a4         # offset = v43a4_JT1-indexer
0406:    rlc                # offset = index*2
0407:    add  l             # offset = index*2 + 0E
0408:    mov  l,a           # [L] := index*2 + 0E
0409:    mov  a,m           # [A] bevat higher address part
040A:    inr  l
040B:    mov  l,m           # [L] bevat nu lower address part
040C:    mov  h,a
040D:    pchl               JT1 JUMP!

"   table (index,address) vanaf 040e:
```

```
    0:00-> $0430 init demovars      3:06-> $0800 JT4               6:0c-> $2400   explosion sequence ?
    1:02-> $04ac score blinking     4:08-> $0aea starfield scroll?  7:0e-> $244c   ---------- ?
    2:04-> $0515    init vars       5:0a-> $0b60 game/match over handling

    040E:  0430
    0410:  04AC 0515 0800 0AEA 0B60 2400 244C
"

0515
------------------------------------------------- init? video control palette
041E:    ldax $43a3         # v43a3_PLtoggle
0421:    ani  $01           # alleen bitwaarde #0 is bepalend
0423:    mov  b,a
0424:    ldax $43b8         #  v43b8_staging0-F_videobit1 (#1 particitapes in video control palette)
0427:    ani  $02           # bit #1    color palette high bit (6)
0429:    ora  b             # combineer bit#1 (a) en bit#0 (b) met elkaar
042A:    stax $5000         VIDEO Control Palette
042D:    ret




JT1:0->$040d
-----------------------------------------------------init demo vars?
0430:    lxi  h,$43a4           # v43a4_JT1-indexer
0433:    mvi  m,$01             #               := 01
0435:    inr  l
0436:    mvi  m,$80             # v43a5_JT1counter := $80, in loop JT1:1 to count down
0438:    mvi  l,$a3             # 43a3
043A:    mov  a,m               # [A] := v43a3_PLtoggle
043B:    mvi  m,$00             # v43a3_PLtoggle := 00
043D:    cpi  $02
043F:    rz

0440:    mov  m,a               # v43a3_PLtoggle : = [A]   (default 02 ?)
0441:    dcr  l
0442:    mov  a,m               # v43a2_curplayer
0443:    cpi  $01
0445:    rz

0446:    inr  l
0447:    mov  a,m               # [A] := v43a3_PLtoggle
0448:    ana  a                 #    (Z when a=0, otherwise NZ)
0449:    jz   $04a0             # other palette
044C:    mvi  l,$90             # v4390_PL1numlives
044E:    mov  a,m
044F:    ana  a                 # (Z when a=0, otherwise NZ)
0450:    rz

0451:    mvi  l,$a3             # 43a3
0453:    mvi  m,$00             # v43a3_PLtoggle := 00
0455:    lxi  b,$0100           # [B=01, C=00]
0458:           call $0460     # schakel per karakter over naar ander palette
045B:    ret




0458, 04A7, 0b91
-----------------------------------------------------schakel per karakter over naar ander palette
0460:    lxi  h,$5000           #Video Control write-only (palette bit#0)
0463:    lxi  d,$4320           # [DE] := schermpositie linksboven  scherm A
(046e,0477,047e)
0466:    mov  m,b               #|  zet de palette op waarde B (0455=01)
0467:    ldax d                 # vervang hetzelfde karakter
0468:    mov  m,c               #|  zet de palette op waarde C (0455=00)
0469:    stax d                 # zodat de bitmap ververst wordt
046A:    inr  e                 # (1e run 21, 2e:22, 3e:23, 4e run stop ivm and 03)
046B:    mov  a,e
046C:    ani  $03               # de eerste 3 regels 01/10/11 waarden
046E:    jnz  $0466
```

```
0471:    mov  a,e              # (24)
0472:    ani  $f0              # (24 and f0 -> 20) truukje voor bovenste positie  only high nibble
0474:    sui  $20              # (20-20 geeft 0) kolom naar rechts
0476:    mov  e,a
0477:    jnc  $0466

047A:    dcr  d                # [43..3F>
047B:    mov  a,d
047C:    cpi  $3f              # scherm is in range 40.... dus 3F
047E:    jnz  $0466

0481:    lxi  d,$4380          # PL1 High score
0484:    mov  m,b
0485:    ldax d
0486:    mov  m,c
0487:    stax d
0488:    inr  e
0489:    mov  a,e
048A:    cpi  $b8
048C:    jnz  $0484
048F:    lxi  d,$4bc0          #BIRDPOS of ?
(049a)
0492:    mov  m,b
0493:    ldax d
0494:    mov  m,c
0495:    stax d
0496:    inr  e                # 4bc1, 4bc2, ..
0497:    mov  a,e
0498:    cpi  $00
049A:    jnz  $0492
049D:    ret




0449
------------------------------------------------------other palette
04A0:    mvi  l,$a3            # 43a3
04A2:    mvi  m,$01            # v43a3_PLtoggle := 01
04A4:    lxi  b,$0001          # b=00 c=01
04A7:          call $0460      # schakel per karakter over naar ander palette
04AA:    ret



JT1:1->040d
------------------------------------------------------score blinking
#  stay in JT1:1 until v43a5_JT1counter reaches zero
04AC:    lxi  h,$43a5          # v43a5_JT1counter
04AF:    dcr  m                #    := --1
04B0:    mov  a,m
04B1:    dcr  l                # v43a4_JT1-indexer
04B2:    mvi  m,$02            #          := 02    preset to another JT1 loop
04B4:    ana  a                #                   |
04B5:    rz                    #                   will be overruled if counter not zero


04B6:    mvi  m,$01            # v43a4_JT1-indexer := 01    reset JT1 loop
04B8:    cpi  $7f              # v43a5_JT1counter waarde is 7f?
04BA:    jz   $07f0            # ja -> single shot execute of update videoscroll

#  reset v439ab_16bits_counter counter
04BD:    mvi  l,$9a            # 439a
04BF:    mvi  m,$00            #
04C1:    inr  l                #
04C2:    mvi  m,$00            #   v439ab_16bits_counter := 0000
04C4:    ani  $08              #
04C6:    jnz  $04e6            # skip printing the HUD score line again
```

```
#  each countdown cycle of 8 execute following (blank score)
04C9:           call $06e8      #   HUD 1e regel afdrukken
04CC:    nop                    #
04CD:    lxi  h,$43a3           #    v43a3_PLtoggle
04D0:    mov  a,m               #
04D1:    ana  a                 #
04D2:    mvi  l,$83             #    4383
04D4:    lxi  d,$4261           #    videoscherm PL1 score rechterkant
04D7:    jz   $04df             #
04DA:    mvi  l,$87             #    4387
04DC:    lxi  d,$4021           #    videoscherm PL2 score rechterkant
04DF:    mvi  b,$06             #
04E1:           call $00c4      #   plot
04E4:    ret



04c6
-------------------------------------------------------per 8 cycles hide blink score
04E6:    lxi  h,$43a3           #  v43a3_PLtoggle
04E9:    mov  a,m               #    [A] := v43a3_PLtoggle
04EA:    ana  a                 #               (Z when a=0, otherwise NZ)
04EB:    lxi  d,$4261           #           screenpos rightside PL1 score
04EE:    jz   $04f4             #
04F1:    lxi  d,$4021           #           screenpos rightside PL2 score
(04ee)
04F4:    mvi  b,$06             #           6 karakters
04F6:           call $04fb      #           hide score
04F9:    ret



04f6
-------------------------------------------------------maak de score onzichtbaart
(0502)
04FB:    mvi  a,$00
04FD:    stax d
04FE:           call $0210      #   cursor met 1 kolom positie naar links
0501:    dcr  b
0502:    jnz  $04fb
0505:    ret



052C
-------------------------------------------------------clear and init vars (4392-4398)
0506:    lxi  h,$4392               # 4392
0509:    mvi  b,$06
050B:           call $05d8          # CLEAR memory [HL=4392, HL+B=06]
050E:    ldax $4b50                 # v4b50_smallbird-romlists
0511:    stax $4394                 # v4394_4b50copy
0514:    ret



JT1:2->040d
-------------------------------------------------------init vars?
0515:           call $041e
0518:    lxi  h,$43a4               # v43a4_JT1-indexer
051B:    mvi  m,$03                 #           := 03   JumpTable 4
051D:           call $0580          # init vars [43ab .. 43b6]
0520:           call $0547          # init vars [43c0 .. 45e3]
0523:           call $09a0          # init enemy fleet vars
(21d7)
0526:           call $0532          # SETUP enemy fleet
0529:           call $0a6c          # Modify wingformation?
052C:           call $0506          # clear and init vars
052F:    jmp  $32b0                 # clear & init vars
```

```
0526
-----------------------------------------------------SETUP enemy fleet
0532:    lxi  h,$4b50          # v4b50_smallbird-romlists
0535:    mvi  b,$a0
0537:           call $05d8     # CLEAR memory [HL=4b50, HL+B=a0]
053A:           call $05ec     # setup appearance of enemyfleet
053D:           call $0650     # initial sound state setup for enemyfleet
0540:           call $0610     # positions enemyfleet based on stage
0543:    ret



0520
-----------------------------------------------------init vars [43c0 .. 43e0 - 45e3]
0547:    lxi  h,$0560          # rom startbase
054A:    lxi  d,$43c0          # var startbase (special object slot #0)
054D:    mvi  b,$20            #    20 bytes (0560 tot 0580) -> (43c0 tot 43e0)
054F:           call $05e0     #   copy [B] bytes van rom[HL] naar var[DE]
0552:    lxi  h,$43e0          #   vanaf 43e0
0555:    mvi  b,$203           #    tot 45e4
0557:           call $05d8     #   CLEAR memory [HL, HL+B]
055A:    ret


"  SPEC OBJ
          43c0         43c4         43c8         43cc
    0560:  0C 10 64 D8  00 50 00 D0  00 50 00 D0  00 58 00 20
    0570:  00 58 00 20  00 58 00 20  00 58 00 20  00 58 00 20
          43d0         43d4         43d8         43dc
"

(0114) 051D
-----------------------------------------------------init vars [43ab .. 43b6]
  initieer vars met voorgedefineerde waarden uit rom
0580:    lxi  h,$0598       # initial vars base
0583:    ldax $43b8         #  v43b8_staging0-F_videobit1 (used for table selection)
0586:    ani  $0f           # index [0..F]
0588:    add  l             # offset berekenen, offset = m[0598+index]
0589:    mov  l,a
058A:    mov  l,m           # bepaling [L] vb A8  *(0598+0)->a8
058B:    mvi  h,$05         # 05xx verwijzing  vb 05a8
058D:    lxi  d,$43ab       # startlocatie van te vullen variabelen
0590:    mvi  b,$0c
0592:       call $05e0      # copy [B=c] bytes van rom[HL=05xx] naar var[DE=43ab]
0595:    ret

"
        index  in 43b8:    0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f
     0598:                 A8 A8 C0 C0 A8 A8 A8 A8  B4 CC B4 B4 A8 A8 A8 A8

    12 vars    43ab 43ac 43ad 43ae 43af 43b0 43b1 43b2 43b3 43b4 43b5 43b6   (43b2:43b3)
     05a8:       80   7F   00   00   40   3F   00   1C   00   FF   FF   FF   (1c00 starfield1)
     05b4:       60   5F   01   02   30   2F   00   1C   00   C0   FF   FF   (      ''       )
     05C0:       80   7F   03   04   40   3F   00   1F   00   A0   FF   FF   (1f00 starfield2)
     05cc:       60   60   05   06   50   30   00   1D   00   48   FF   FF   (1d00 boss-ship)
  43ab,43af -> timed scrolltriggervalues
  43b2/43b3 -> starfield location
  43ac,43ad,43ae,
  43b0/43b1 -> ?
  43b4,43b5,43b6
"

0158, 050b, 0537, 0557, 32b5, 32bc, 32cd
------------------------------------------------------ CLEAR memory tussen [HL, HL+B]
05D8:    xra  a             # snel A op 0 zetten   {eor a (1+1=0) dus effectief A := 0}
(05d9)
05D9:    mov  m,a           # *(HL++) = 0
05DA:    inx  h
05DB:    dcr  b             # loopcount
05DC:    jnz  $05d9
05DF:    ret
```

```
054f, 0592, 32ed, 3991, 3998, 39a2, 39e3
-----------------------------------------------------copy [B] bytes van rom[HL] naar var[DE]
(05e5)
05E0:    mov  a,m          # vb 05a8->80
05E1:    stax d            # vb 43ab
05E2:    inx  h
05E3:    inx  d
05E4:    dcr  b
05E5:    jnz  $05e0
05E8:    ret



053A
-----------------------------------------------------setup appearance of enemyfleet (WX  part wingdata [WXYZ])
  DE -> *(base1500+offset) 086c/0960
05EC:    lxi  h,$1500          # preset wingdata info table
05EF:    ldax $43b8            #  v43b8_staging0-F_videobit1  (contributes to wingdata WX part)
05F2:    ani  $0f              # 0000.xxxx beperk tot 16 entries
05F4:    rlc                   # 000x.xxx0 2 bytes per entry offset
05F5:    add  l                # 1500[offset]
05F6:    mov  l,a
05F7:    mov  d,m              # DE gets value based on offset
05F8:    inx  h                # as we see
05F9:    mov  e,m              # this will be 086c or 0960 for WX

"
          0      2      4      6      8      a      c      e
    1500:  [086C] [0960] [086C] [0960] [086C] [0960] [086C] [0960]  WX wingformation
    1510:  [086C] [0960] [086C] [0960] [086C] [0960] [0960] [0960]  byte pairs
          10     12     14     16     18     1a     1c     1e
"

0842
------------------------------ set for remaining enemies [WXyz] WX part (animation index) to [DE]
05FA:    lxi  h,$4b70          # v4b70_wingformation-base
05FD:    ldax $43ba            # v43ba_smallbird_fleetsize
0600:    mov  b,a              #   in [B] as loop count
0601:    ana  a                # (Z when a=0, otherwise NZ)
0602:    rz                    # v43ba_smallbird_fleetsize == 0

(060a) # loop for next remaining bird
#  enemies left to fill, HL points to a smallbird wingformation to be set
# [DE] contains the new WX part, for example [08 6c]
# startcondition 18 6c, goes to 08 6c making the bird object depicted by 6c (see comments at @1400 to disappear)
0603:    mov  m,d              #       [WXyz] setting W part
0604:    inr  l
0605:    mov  m,e              #       [WXyz] setting X part
0606:    inr  l                #
0607:    inr  l                #
0608:    inr  l                #   HL+4 for next loop, start of next tupple
0609:    dcr  b                # loopcounter --
060A:    jnz  $0603            # loop until [B==0]
060D:    ret



0540
-------------------------------------------------- initial positioning enemyfleet based on stage
#(fill YZ part wingdata [wxYZ])
0610:    lxi  h,$063a          # see at 063a
0613:    ldax $43b8         # v43b8_staging0-F_videobit1  (contributes to wingdata YZ part)
0616:    rrc
0617:    ani  $0f              # maintain low-end nibble
0619:    add  l
061A:    mov  l,a              #3a + stage
```

```
061B:    nop
061C:    nop
061D:    nop
061E:    mov  l,m              # stage == 00 -> 60, ....
061F:    mvi  h,$15            # 15[60],15[40],15[e0]..
0621:    lxi  d,$4b72          # YZ part of wingdata [WXYZ] per object, W starting at 4b70
0624:    ldax $43ba            #   v43ba_smallbird_fleetsize
0627:    mov  b,a
0628:    ana  a                # (Z when a=0, otherwise NZ)
0629:    rz                    # return if fleetsize==0
(0635) ## loop for next remaining bird
062A:    mov  a,m
062B:    stax d                # Y part (ex. 4b72,4b76,...)
062C:    inx  h
062D:    inx  d
062E:    mov  a,m              # Z part (ex. 4b73,4b77,...)
062F:    stax d
0630:    inx  h
0631:    inx  d
0632:    inx  d
0633:    inx  d                # next enemy YZ part
0634:    dcr  b                # count down to zero
0635:    jnz  $062a
0638:    ret

"                                    determine offset based on v43b8_staging0-F_videobit1
    063a:                             60 40 E0 E0 E0 E0
    0640:  FF FF C0 A0 80 80 80 80 FF FF FF FF FF FF FF FF

     for  1500 + offset:  which enemy smallbird wave # data base
              00=1560, 01=1540, 02=15e0, 03=15e0, 04=15e0, 05=15e0  (06:15ff,07:15ff)
              08=15c0, 09=15a0, 0a=1580, 0b=1580, 0c=1580, 0d=1580  (0e:15ff,0f:15ff)

    1500 + [offset] to fill wingdata [WXYZ], in which YZ part is used to calculate *(0xa00+Yy)+Zz for the sprite data
     50-> 0101.000 -> 0.1010(000) -> entry #0a -> column 10 of screen
     +20-> 0010.0000->0.0100(000)->               row     4 of screen
     xy(0,0) origin is at topleft, (4,10) means 5th column from the left and then 11 rows down (1th row = row0)


offset n->
01->1540:  5020 7020 6028 6038  5040 7040 4038 8038   (10,4) (14,4) (12,5) (12,7)  (10,8) (14,8) (8,7)  (16,7)
    1550:  3030 9030 2038 A038  1848 A848 6048 6058   (6,6)  (18,6) (4,7)  (20,7)  (3,9)  (22,9) (12,9) (12,11)
 phoenix mega bird, fifth level

                     1 1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
4                    x       x
5                        x
6             x                      x
7         x       x       x       x      x
8                    x       x
9       x                x                   x
10
11                       x
12


00->1560:  6048 6058 4858 7858  3850 8850 2848 9848   a(12,9) b(12,11) c(9,11) d(15,11)  e(7,10) f(17,10) g(5,9) h(19,9)
    1570:  1840 A840 1830 A830  2828 9828 3820 8820   i(3,8)  j(21,8)  k(3,6)  l(21,6)   m(5,5)  n(19,5)  o(7,4) p(17,4)
   small birds 'W' wave, first level

                     1 1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
```

```
4              o                   p
5          m                           n
6       k                                 l
7
8       i                                 j
9           g            a            h
10             e                   f
11                 c     b     d
12



offsets 0a,0b,0c,0d->
    1580:  6020 5020 7020 4028  8028 3030 9030 2038   (12,4) (10,4)  (14,4)  (8,5)    (16,5) (6,6)   (18,6) (4,7)
    1590:  A038 6058 5058 7058  4058 8058 3058 9058   (20,7) (12,11) (10,11) (14,11)  (8,11) (16,11) (6,11) (18,11)

                     1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
4                    x   x   x
5                x               x
6            x                       x
7         x                              x
8
9
10
11           x   x   x   x   x   x  x
12



09->15A0:  6020 5028 7028 4030  8030 3038 9038 2040   (12,4) (10,5)  (14,5)  (8,6)    (16,6) (6,7)   (18,7) (4,8)
    15B0:  A040 6058 5058 7058  4050 8050 3048 9048   (20,8) (12,11) (10,11) (14,11)  (8,10) (16,10) (6,9)  (18,9)

                     1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
4                        x
5                    x       x
6                x               x
7            x                       x
8         x                              x
9            x       x   x   x       x
10               x               x
11



08->15C0:  6058 5050 7050 6048  4048 8048 5040 7040   (12,11) (10,10) (14,10) (12,9)  (8,9) (16,9) (10,8) (14,8)
    15D0:  4038 8038 3030 9030  2028 A028 1020 B020   (8,7)   (16,7)  (6,6)   (18,6)  (4,5) (20,5) (2,4)  (22,4)

                     1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
4     x                                    x
5         x                           x
6             x                       x
7                 x               x
8                     x       x
9                 x       x       x
10                    x       x
11                        x
12
```

```
offsets 02,03,04,05->
    15E0:  6020 5028 7028 4030  8030 3038 9038 2040  (12,4) (10,5) (14,5) (8,6)   (16,6) (6,7) (18,7) (4,8)
    15F0:  A040 6020 5028 7028  4030 8030 3038 9038  (20,8) (12,4) (10,5) (14,5)  (8,6) (16,6) (6,7)  (18,7)
   smallbird 9-wave during big boss (below: X = 2 birds on same position, + = single bird)

                     1 1 1 1 1 1 1 1 1 1 2 2 2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0
1
2
3
4                        X
5                     X       X
6                  X              X
7               X                     X
8         +                              +
9
10
11
12

"

053D
------------------------------------------------------ initial rom setup for enemyfleet
0650:    lxi  h,$1520       # base (in ROM)
0653:    ldax $43b8         #  v43b8_staging0-F_videobit1  (contributes to sound state)
0656:    ani  $0f           # (lownibble gebruikt als index)
0658:    rlc                #   *2
0659:    add  l
065A:    mov  l,a           # offset will be $1520+index*2  (16bits)
065B:    mov  d,m
065C:    inx  h
065D:    mov  e,m           # DE := 1000  = object alive? 00 00  = object deleted?
065E:    lxi  h,$4b50       # v4b50_smallbird-romentry
0661:    ldax $43ba         # v43ba_smallbird_fleetsize
0664:    mov  b,a
0665:    ana  a
0666:    rz                 # als v43ba_smallbird_fleetsize == 0

#  enemies left...
(066c)
0667:    mov  m,d           # fill v4b50_smallbird-romlistssound  with  D-val  4b50, 4b52, 4b54, .. 4b5e
0668:    inr  l
0669:    mov  m,e           # fill v4b51  with E-val                          4b51, 4b53, .........4b5f
066A:    inr  l
066B:    dcr  b
066C:    jnz  $0667         # until itemcount [B] ==0
066F:    ret

"
smallbirds romlists pointers data (gaat naar 4b50-4b6f)
    1520:  1000 1000 1000 1000 1000 1000 1000 1000
    1530:  1000 1000 1000 1000 1000 1000 1000 1000
"



??
------------------------------------------------------
    0670:    lxi  h,$43b1       # v43b1_16bit_romptr_low
    0673:    mov  b,m
    0674:    mvi  l,$b9         # v43b9_videoscroll
    0676:    mov  c,m
    0677:    mov  a,c
    0678:    sub  b
    0679:    mov  m,a

06F0, 22b4, (24ec)
-------------------------------------------------------scroll background and plot starfield (?)
067A:    lxi  h,$43b9       # v43b9_videoscroll                       vb 00
```

```
067D:    mov  a,m
067E:    dcr  m             #   v43b9_videoscroll -=1              vb FF
067F:    stax $5800         VIDEO Scroll REg  #:= [A]               vb 00
0682:    ani  $07           # 0000.0111 (7/8? cycles tussenpoze)
0684:    rnz                # er is 1 pixel gescrolled 'naar onderen' (rechts<--ROT90!)

"  video scroll register in bij ingang 00
  plot 1 regel van een background starfield 2e rij bovenaan van het scherm
     elders: scrollen + afdrukken van PHOENIX tekst in de vorm van smallbird variaties (011f->0196)
  (43b2:43b3) pointer naar starfield rom locatie met standaardwaarde(1C:00), zie code vanaf $0580
  copy slag van [HL] naar [DE]   HL wordt bepaald via (43b2:43b3) pointer
"
0685:    lxi  b,$2047       # B=kolombreedte C=stopping case 47 ok is [4b t/m 48] zie [D]
0688:    lxi  d,$4b21       # beginpositie background starfield loc(31,1)
068B:    mov  a,m           # v43b9_videoscroll (was net --1 gedaan)  vb FF
068C:    rrc                #    0xxx.xxxx   delen door 8
068D:    rrc                #    00xx.xxxx   dus per 8 pixels
068E:    rrc                #    000x.xxxx   een nieuwe starline plotten
068F:    ani  $1f           #   vb 1F
0691:    add  e             #   vb 1F+21  (of bij 00, 00+21)
0692:    mov  e,a           #   vb 40  (dan DE=4b40) 1 postitie verder dan rechtsonder
0693:    mvi  l,$b2         # 43b2    hi-byte starfield rom pointer
0695:    mov  a,m           #       gezet in routine 0580, bvb 1C/1F/1D
0696:    inr  l             # 43b3    vasthouden starfield low-byte offset
0697:    mov  l,m           # 43b3  default op 00 ==> 4300
0698:    mov  h,a           # 1c00    [vb A=1c]
(069f,06a5)
0699:    mov  a,m           # 1c00->00 (ZZ)
069A:    stax d             # [DE] := ZZ        starfield background plot
069B:    inr  l             # vb 1c01
069C:    mov  a,e           # \
069D:    sub  b             # | E - 20      kolomlengte aftrekken dus 1 kolom rechts
069E:    mov  e,a           # /
069F:    jnc  $0699
06A2:    dcr  d             # 4b->4c->...47
06A3:    mov  a,d
06A4:    cmp  c             # bij 47 stoppen (backgroundscherm gaat tot 4800)
06A5:    jnz  $0699
06A8:    mov  a,l
06A9:    stax $43b3         # vasthouden 1Cxx [L] starfield XX rom locatie
06AC:    ret

"
    starfield rom start locations 1c00/1d00/1f00 (subroutine 0580)
    1C00:  [00] 01 00 06 00 02 03 04 00 01 00 08 00 02 03 04
    1D00:  0C   0D 0C 0F 07 07 01 00 00 4C 4D 4E 4F 4F 4E 4D
    1F00:  00   00 00 01 00 00 00 02 00 00 00 00 03 00 00 00
"



(06f6)
------------------------------------------------------background plot on timed trigger?
06B0:    lxi  h,$43ab          # v43ab_scrolltrigger
06B3:    ldax $43b9            # v43b9_videoscroll
06B6:    mov  c,a              #       backup
06B7:    cmp  m                #
06B8:    rnz

#  v43ab_scrolltrigger == v43b9_videoscroll
06B9:    mov  a,m              # v43ab_scrolltrigger
06BA:    inr  l
06BB:    add  m
06BC:    dcr  l
06BD:    mov  m,a
06BE:    inr  l
06BF:    inr  l
06C0:    inr  m
06C1:    mov  b,m
06C2:    inr  l
06C3:    inr  m
```

```
06C4:    mov  a,m
06C5:    lxi  h,$1e20          # -> L1 base for backgroundscreenpointers
06C8:    ani  $1f
06CA:    add  l                # HL:=1e20[A]   [0<=A<=1F]
06CB:    mov  l,a
06CC:    mov  d,m              # waarde in D       vb 49
06CD:    adi  $20              # nieuwe kolom?
06CF:    mov  l,a              # HL:=1e20[A+20]
06D0:    mov  e,m              # waarde in E       vb A0,  geeft dus DE=49a0
06D1:    mov  a,c              # v43b9_videoscroll
06D2:    rrc                   #           0xxx.xxxx
06D3:    rrc                   #           00xx.xxxx
06D4:    rrc                   #           000x.xxxx
06D5:    ani  $1e              #   reikwijdte van een kolom (000x.xxx0)
06D7:    add  e                # + 1e rij pointer
06D8:    adi  $02              # + 2
06DA:    mov  e,a              # nieuwe pointer vb 49b2
06DB:    lxi  h,$1e60          # -> L2
06DE:    mov  a,b
06DF:    ani  $1f
06E1:    add  l
06E2:    mov  l,a
06E3:    mov  l,m
06E4:       call $07dc
06E7:    ret

"
L1: 1E20:  49  48  4A  4B  4A  49  4A  49  48  4A  48  49  4B  48  4A  48
    1E40:   A0  60  40  00  E0  C0  C0  60  80  20  60  40  20  40  00  80

    1E30:  4A  49  4B  49  4B  4A  49  48  49  49  4A  4A  48  49  4A  48
    1E50:   40  00  20  E0  00  60  00  A0  E0  20  80  00  C0  80  A0  E0

    (49a0,4860,4a40 ...) 32 background screenpointers (verwijzingen naar 1e rij)

L2: 1E60:  00 04 08 0C 10 14 18 1C 00 08 10 18 04 0C 14 1C
    1E70:  00 0C 18 04 04 1C 08 14 00 10 04 14 08 18 0C 1C

    1E80:  10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
    1E90:  10 12 14 16 18 1A 1C 1E 11 13 15 17 19 1B 1D 1F

"


04c9
------------------------------------------------------ 1e regel van HUD plotten
06E8:    lxi  h,$1800          # 1800:  43 20  linker bovenhoek scherm
06EB:    mvi  c,$01            # 1 regel tekst
06ED:    jmp  $01d0            # PrintLNs

    1800:  (4320) FFFFFFFF 00 13 03 0F 12 05 21 00 00 08    "         SCORE1 H"
    1810:  09 2B 13 03 0F 12 05 00 00 13 03 0F 12 05 22 00    "I-SCORE  SCORE2 "
    ...


(007B,2452,24cb), 0834
------------------------------------------------------
06F0:          call $067a          # scroll background and plot starfield
06F3:          call $2040          # plot on timed trigger?
06F6:    jmp  $06b0                # plot on timed trigger?



0876
------------------------------------------------ spec obj positioning
# 3 slots 43c0/4/8 to be checked
0700:    lxi  b,$43c0          # 43c0      special object slot #0 (SpaceShip)
0703:    lxi  d,$43e0          # 43e0      special objects screenpos table
(0713)
0706:          call $0718      # SHOW/HIDE enemy/ship?
0709:    mov  a,c
```

```
070A:    adi  $04              # 43c4/43c8  (slow/rapid laser fire)
070C:    mov  c,a
070D:    adi  $20              # 43e4/43e8
070F:    mov  e,a
0710:    mov  d,b
0711:    cpi  $ec              #    looping unless 43ec
0713:    jnz  $0706            # loopback for the two remaining entries
0716:    ret

    # 0717: C9         ret



0706, 0a57, 0cdf
-----------------------------------------------------SHOW/HIDE enemy/ship?
0718:          call $0720      # check for ALREADY HIDDEN smallbird / ship /
071B:    jmp  $0740            # check to SHOW smallbird/ ship /

    # 071E: E6 EF      ani  $ef

0718
-----------------------------------------------------check if ojbect as HIDE bit set in [Wxyz] for W
0720:    ldax b                #  [via 0718-> 0706:B=43c0, 0a57:B=4b70, 0cdf:B=43cc]
0721:    mov  h,a              #      (DE points to WINGPOS 4bb0/43E0)
0722:    ani  $10              # base[0] == 0001.0000 bit #4 HIDE set (alterively bit#3 lefthanded)?
0724:    rz                    #  NOT set, skip the proceeding 'hide' object drawing


------------------------------------------------- JUMP TABLE 2 #hide (ploting 00's) objects
# example BC=43c0, DE=43e0
# object [WW xx yy zz ]
# bit #4 of WW is the HIDE (by plotting zeros) indicator
# bits #765 determines the JT2 index entry
0725:    mov  a,h       #       [H<>10]
0726:    ani  $ef       #       yyy0.xxxx   clear out bit#4  example: 9c->8c
0728:    stax b         #  43c0 update  (like :- 29)
0729:    rlc            #       xx0x.xxxy      bits #765 is offset to the JT2 entry
072A:    rlc            #       x0xx.xxyy
072B:    rlc            #       0xxx.xyyy
072C:    ani  $07       #       0000.0yyy
072E:    adi  $38       #       0011.1yyy
0730:    mov  l,a
0731:    mvi  h,$07     #  HL=07..
0733:    mov  l,m       # [00:0738=63-->0763] [01:0739=79-->0779]
0734:    pchl           JP2 JUMP!

" JUMP TABLE 2 (for clearing out the object by ploting '00' values)
  yyy=     0    1   2   3    4   5  6  7
    0738: [63] [79] FF [9E] [BE] FF FF FF
        JT2
        00=> 07 63 -> HIDE (plot 00) enemy (small)bird (1byte)  = JT2:0
        01=> 07 79 -> HIDE (plot 00 00) (small)bird (2bytes)    = JT2:1
        02=> 07 FF ->  NOT VALID
        03=> 07 9e ->    JT2:3
        04=> 07 be ->    JT2:4
        05=> 07 FF ->  NOT VALID
        06=> 07 FF ->  NOT VALID
        07=> 07 FF ->  NOT VALID
"

071b   (proceed after Jump)
-----------------------------------------------------check if ojbect as SHOW bit set in [Wxyz] for W
0740:    ldax b             # fleet base [0706:B=43c0 SpaceShip, 0a57:B=4b70 wing, 0cdf:B=43cc EnemyBomb]
0741:    mov  h,a           #      (DE points to WINGPOS 4bb0/43E0)
0742:    ani  $08           # bit #3 SHOW set?
0744:    rz                 # NO, skip drawing

------------------------------------------------- JUMP TABLE 3 (-->076d/07d2)  #plot/show object
# object [WW xx yy zz ]
# bit #3 of WW is the SHOW indicator
# bits #210 determines the JT3 index entry
```

```
0745:    mov  a,h
0746:    ani  $07           # 0000.0xxx
0748:    mov  h,a           #
0749:    rrc                # x000.00xx
074A:    rrc                # xx00.000x
074B:    rrc                # xxx0.0000
074C:    ora  h             # xxx0.0hhh ?
074D:    ori  $18          #   set bit #4 and #3   xxx1.1hhh (HIDE+SHOW)
074F:    stax b            #  43c4 (spec.obj1.val0 = laser of spaceship) := 39
0750:    inx  b            #
0751:    mov  a,h          #   0000.0xxx
0752:    adi  $5b          #    base is 5b voor 075b
0754:    mov  l,a
0755:    mvi  h,$07        # 075b
0757:    mov  l,m          # new offset := value of [07.L]
0758:    pchl               JT3 JUMP! # PC:= HL (00->075B=6D->076d, 04->075F=D2->07d2)

"   JUMP TABLE 3
     lookup, gebruik waarde van 075b[index] voor [L] in HL=07xx
                0    1   2   3    4   5  6  7
        075b:  [6D] [88] FF [AA] [D2] FF FF FF

            00=075b    04=075f
                  |          |
    075b:         6D 88 FF AA D2
                      |    |
                 01=075c    03=075e:AA
        JT3
        00=>076d        -> PLOT opening sequence smallbird (1 byte row size)
       [02,05,06,07     => 07FF -> NOT]
        01=>0788        -> SHOW (plot) enemy smallbird (2bytes [L R] on same row size)
        03=>07aa        -> plot 2 byte diving smallbird [T] [B] format on screen
        04=>07d2        -> plotten the spaceship itself
"


JT2: Jump Table 2 entry 00 (0734)
-------------------------------------------------------HIDE (plot 00) enemy smallbird (1byte)
0763:    xchg
0764:    mov  d,m
0765:    inx  h
0766:    mov  e,m
0767:    dcx  h
0768:    xra  a             # 00
0769:    stax d             # plot 00
076A:    xchg
076B:    ret



    # 076C: EB         xchg

JT3: Jump Table 3 entry00 (0758)
-------------------------------------------------------PLOT opening sequence smallbird (1byte row size)
076D:     xchg              # example smallbird pointers (DE=4bc8,HL=076D JR3)
076E:     inx  h
076F:     inx  h
0770:     mov  d,m          # (HL=4bc8+2  4bca)
0771:     inx  h               # 4bc9
0772:     mov  e,m
0773:     ldax b               # enemy bird
0774:     stax d               # PLOT opening sequence smallbird on videoscreen [DE pointer to]
0775:     dcx  b
0776:     ret



    # 0777: 12         stax d
    # 0778: 23         inx  h

JT2: Jump Table 2 entry 01 (0734)
```

```
-----------------------------------------------------HIDE (plot 00 00) smallbird (2bytes in row size)
0779:    xchg
077A:    mov  d,m
077B:    inx  h                 #  (WINGPOS 4bb0...)
077C:    mov  e,m               # DE screenpos
077D:    dcx  h                 #  (WINGPOS 4bb0...)
077E:    xra  a                 # 00
077F:    stax d
0780:           call $0217      #   cursor met 1 kolom positie naar rechts
0783:    xra  a                 # 00
0784:    stax d                 # clean screenpos
0785:    xchg
0786:    ret

    # 0787: 23         inx  h


JT3: Jump Table 3 entry01 (0758)
-----------------------------------------------------SHOW (plot) [L R]  smallbird (2bytes in row)
#  wingdata [WXYZ] wingpos [DE], X is base index 1400[X] for a smallbird object
#  W bitvalues: #4=1 -> HIDE object, #3=1 -> SHOW object
0788:    xchg              # 4bxx (wingdata position)
0789:    inx  h            #   volgende enemy schermpos
078A:    inx  h            #   16 bits dus twee bytes opschuiven
078B:    mov  d,m
078C:    inx  h
078D:    mov  e,m          # schermpos in DE laden
078E:    ldax b            #   X van formatiestructuur bevat lowbyteindex
078F:    mov  l,a          #   $14[X]  X is  start index for smallbird LEFTside char
0790:    mvi  h,$14        #   in 14xx Animation sequence smallbird+ship
0792:    mov  a,m          #   vb X=22 $1422 -> [62 63] plot at 424a/422a
0793:    stax d            # schermpos(d) plot  (linkerkant van smallbird?)
0794:    inx  h            #   14[X+1] -> next byte for smallbird RIGHTside char
0795:       call $0217     #   cursor met 1 kolom positie naar rechts
0798:    mov  a,m
0799:    stax d            # schermpos(d) plot  (rechterkant van smallbird?)
079A:    dcx  b
079B:    ret


JT2: Jump Table 2 entry 03 (0734)
---------------------------------- HIDE smallbird top down bytes [T][D] with (plot 00 00)
079E:    xchg
079F:    mov  d,m
07A0:    inx  h
07A1:    mov  e,m
07A2:    dcx  h
07A3:    xra  a         #plot 00 as char on screen
07A4:    stax d
07A5:    inx  d
07A6:    stax d
07A7:    xchg
07A8:    ret


JT3: Jump Table 3 entry03 (0758) ??
-----------------------------------------------------diving smallbirds [T] [B] bytes plot
# see anim sequence range at 1460 below
#   ex [DE=4bb0], [BC=4b81]
07AA:    xchg
07AB:    inx  h
07AC:    inx  h
07AD:    mov  d,m                #4bc2
07AE:    inx  h
07AF:    mov  e,m
07B0:    ldax b                  #4b81
07B1:    mov  l,a
07B2:    mvi  h,$14              # 14xx Animation sequence location SHIP/.. 1460/1462/1464/1466
```

```
07B4:    mov  a,m              # 4b81 = 5c ->  1458 [69 00]
07B5:    stax d                    #plot smallbird at screen
07B6:    inx  h                    #  [Top]
07B7:    inx  d                    #  [Bottom]
07B8:    mov  a,m                  #      1461/1463/1465/1467
07B9:    stax d                    #plot   one below
07BA:    dcx  b                    #4b80
07BB:    ret

    # 07BC: 23         inx  h
    # 07BD: 13         inx  d



JT2: Jump Table 2 entry 04 (0734)
---------------------- HIDE 4 bytes (spaceship) object [2x2] [TL][TR] [BL][BR] (with plot 00 00)
# after xchg HL points to the screenpointer table 43e0
07BE:    xchg
07BF:    mov  d,m       # screen pos hi byte
07C0:    inx  h
07C1:    mov  e,m       # screen pos low byte
07C2:    dcx  h
07C3:    xra  a             # plot 00
07C4:    stax d         # clear area
07C5:    inx  d
07C6:    stax d         # clear area
07C7:       call $0217      #   cursor met 1 kolom positie naar rechts
07CA:    xra  a             #
07CB:    stax d         #clear area
07CC:    dcx  d
07CD:    stax d         #clear area
07CE:    xchg
07CF:    ret



JT3: Jump Table 3 entry 04 (0758)  plotting the SpaceShip animation seq.
------------------------------------------------------------------------
07d2:   xchg
07D3:    inx  h
07D4:    inx  h         # 43e2 new SS-shippos (2)
07D5:    mov  d,m
07D6:    inx  h
07D7:    mov  e,m
07D8:    ldax b         # 43c1  <-- index for ship-anim (0/4/../1c)
07D9:    mov  l,a
07DA:    mvi  h,$14         # 14xx Animation sequence location SHIP/..

" anim sequence for SpaceShip (take 90 degrees rotatin in consideration)
     format [30 31]                                        in memory line 0:  31 41  (byte #2, #3)
     (2x2)  [40 41]                                                   line 2:  30 40, (byte #0, #1)
   43c1==   0             4             8             c
    1400:  [30 40 31 41] [32 42 33 43] [34 44 35 45] [36 46 37 47]
   43c1==   10            14            18            1c
    1410:  [38 48 39 49] [3A 4A 3B 4B] [3C 4C 3D 4D] [3E 4E 3F 4F]

    the animation depics the spaceship  like a pumping rythm (smaller, normal, bigger) format
"

06E4
----------------------------------PLOT 4 bytes of object [2x2] format
07DC:    mov  a,m
07DD:    stax d
07DE:    inx  h
07DF:    inx  d
07E0:    mov  a,m
07E1:    stax d
07E2:    inx  h
07E3:    dcx  d
07E4:       call $0217      #   cursor met 1 kolom positie naar rechts
07E7:    mov  a,m
```

```
07E8:    stax d
07E9:    inx  h
07EA:    inx  d
07EB:    mov  a,m
07EC:    stax d
07ED:    dcx  b
07EE:    ret



04ba
-----------------------------------update videoscroll, executed when v43a5_JT1counter is 7F
07F0:    ldax $43b9         # v43b9_videoscroll
07F3:    stax $5800         VIDEO Scroll Reg
07F6:       call $0380      #   schoon foreground behalve HUD (1e 3 regels)
07F9:    jmp  $041e

"  07FC: FF FF FF FF"

JT1:3->$040d
-----------------------------------------------------JUMP TABLE 4  scroll anims
0800:    lxi  h,$0814           # see jumptable below
0803:    ldax $43b8             #  v43b8_staging0-F_videobit1  (as gamestage index 0-F JT4 jump)
0806:    rlc                    #                          (make a 16 bits pointer index  0f->1e)
0807:    ani  $1e               # mask only low-nibble
0809:    add  l                 # add base table pointetr 0814
080A:    mov  l,a
080B:    mov  a,m
080C:    inr  l
080D:    mov  l,m
080E:    mov  h,a
080F:    pchl                   JT4 JUMP!

"
0814:            0834 2000 0834 2000 2230 3400
0820:  2230 3400 2230 22B4 22CA 2000 224C 224C
0830:  224C 224C

0840:  5A 08 CD FA 05 CD 50 0A 21 B4 43 7E A7 C0 2E B8
0850:  34 2E A4 36 02 C9 FF FF FF FF

    JumpTable 4: (scroll anim frames)
    00,04       ->0834      =fast row starfield scrolldown
    02,06,16    ->2000      =smallbird fight - enemy wave sequence-step
    08,0c,10    ->2230      =square '*' window anim (center towards border star & clean star anim)
    0a,0e       ->3400      =bigbird   fight  - enemy wave sequence-step
    12          ->22b4      =anim appearing big-boss baseship rowscroll
    14          ->22ca      =anim baseship row scrolldown
    18,1a,1c,1e ->224c      ? in smallbird + bigboss fight  next stage (4->5, 8->9)


"


JT4:{00,04}<-080F     fast row starfield scrolldown
-----------------------------------------------------
0834:       call $06f0             # scroll background and plot
0837:    lxi  h,$43b4              # v43b4_JT4countdown
083A:    dcr  m                    #  '' --1
083B:    mov  a,m                  #
083C:    cpi  $15
083E:    rnc                       # return if countdown still is >= 15

083F:       call $085a             # bepaal wingdata opkomende smallbirds obv countdown (in [DE] obv [A])
0842:       call $05fa             # prepare fleet wingdata met waarde in DE
0845:       call $0a50             # plot enemy fleet?

(22BE)
0848:    lxi  h,$43b4              # v43b4_JT4countdown
084B:    mov  a,m                  #
084C:    ana  a                    # (Z when a=0, otherwise NZ)
```

```
084D:    rnz                        # return if countdown not zero
#---                                counted down
084E:    mvi  l,$b8                 #  v43b8_staging0-F_videobit1   (next stage attribute?)
0850:    inr  m                     #              := ++1    verhoging
0851:    mvi  l,$a4                 # v43a4_JT1-indexer
0853:    mvi  m,$02                 #               := 02    init vars?
0855:    ret



083F
---------------------------------------------------- appearing smallbirds countdown wingdata settings
# bepaal wingdata in DE obv countdown val [A]
# 6c/6d/6e/6f/68 is de smallbirds sequentie als ze opkomen
085A:    lxi  d,$086c       #       DE=086c
085D:    cpi  $11
085F:    rnc                # return if [A >= 11]     (nc gives >=, c gives >)
                            # [A<11]
0860:    mvi  e,$6d         #       DE=086d
0862:    cpi  $0d
0864:    rnc                # return if [A >= 0d]
                            # [A<0d]
0865:    mvi  e,$6e         #       DE=086e
0867:    cpi  $09
0869:    rnc                # return if [A >= 09]
                            # [A<09]
086A:    mvi  e,$6f         #       DE=086f
086C:    cpi  $05
086E:    rnc                # return if [A >= 05]
                            # [A<05]
086F:    mvi  e,$68         #       DE=0868
0871:    ret



2000, 3400
---------------------------------------------------- space ship handling routines
0876:       call $0700      # spec obj positioning
0879:       call $0886      # something todo with space-ship collisioning?
087C:       call $08a0      # spaceship laser shooot handling
087F:       call $09a0      # new positions for spaceship and lasers
0882:       call $097a      # determine new hi/low spaceship horizontal-pixel boundaries
0885:    ret



0879
---------------------------------------------------- appeareance of spaceship anim or collision?
0886:    lxi  h,$43eb   #leftpos above? spaceship screenpos pointer
0889:    mvi  b,$03     #loopcount

0C45
---------------------------------------------------- make copy of word one down
# word copy: b1 b2 b3 b4 -> b3 b4 b3 b4
(0894 b-loop)           # first run example:
088B:    mov  d,m       # 43eb (0886) or 43ff (0c40)
088C:    dcx  h         # 43ea  / 43fe
088D:    mov  e,m       # tmp copy 43ea / 43fe
088E:    dcx  h         # 43e9  /  43fd
088F:    mov  m,d       # *43e9 := *43eb / *43fd := *43ff
0890:    dcx  h         # 43e8  /  43fc
0891:    mov  m,e       # *43e8 := *43ea / *43fc := *43fe
0892:    dcx  h         # 43e7 / 43fb
0893:    dcr  b         #count down from loopcount 03 (0889) or loopcount 05 (0c43) to 1
0894:    jnz  $088b     #b-loop
0897:    ret



087C
---------------------------------------------------- spaceship laser shot handling
08A0:       call $08c4          # barrier and left/right handling
```

```
08A3:    lxi  h,$43c4              #  (special object slot #1  normal laser fire)
08A6:       call $0930          # laser projectile plot
08A9:    ldax $43b8                #  v43b8_staging0-F_videobit1  (determine laser shot speed)
08AC:    ani  $0f                  # maintain low-end nibble
08AE:    cpi  $03                  #  are we at stage 3?
08B0:    rnz
#---                               yes, stage #3 smallbird fight
08B1:    lxi  h,$43c8              #  (special object slot #2  hispeed laser fire)
08B4:       call $0930          # laser projectile plot
08B7:    ret



08A0
--------------------------------------   Do Barrier and LEFT/RIGHT Ship Movement
08C4:    lxi  h,$43c0       #   special object slot #0  (Barrier Status: space-ship INFINITE SHIELD bij 84?)
08C7:    mov  a,m           # Get Barrier Status
08C8:    ani  $08           # ??????????Is barrier already ON?  bit nr #3 set? (hit=1)
08CA:    jz   $0aa0         #no skip.  Yes, so go handle barrier

08CD:    mvi  l,$a6         # v43a6_barrier_repeat_timer
08CF:    mov  a,m
08D0:    ana  a             # Has proper time expired before nextbarrier?
08D1:    jnz  $08ea         # still counting, so don't allow barrier now, skip ahead

08D4:    mvi  b,$80         # Check Barrier Button (bit 7): bit #7 BARRIER (1000.0000)
08D6:       call $00bb      # Check if Barrier input is active, has INput var bit [B] value changed?
08D9:    jz   $08eb         # Barrier not pressed so skip ahead, z=1

----- Barrier button pressed -----
08DC:    mvi  l,$62         # 4362  (v4362_barrier_sound_timer)
08DE:    mvi  m,$40         #  Set Barrier Sound Timer to 64   := 40
08E0:    mvi  l,$c0         # 43c0, Barrier Status  from special object slot #0
08E2:    mov  a,m
08E3:    ani  $f7           # Turn Barrier ON (and clears bit3 to 0)
08E5:    mov  m,a.          # Save Barrier Status
08E6:    mvi  l,$a6         # v43a6_barrier_repeat_timer
08E8:    mvi  m,$ff         #  := FF reset, Don't allow another barrier for FF counts
(08d1)
08EA:    dcr  m             #   --1 , Decrement Barrier Repeat Timer
(08d9)
----- Do LEFT/RIGHT Ship Movement -----
08EB:    mvi  l,$c2         # Ship X Position ,v43c2_SpaceShip-horizPixelpos (spec.obj0.val2)
08ED:       call $0900      # LEFT/RIGHT Ship Movement, LEFT+RIGHT button&boundary handling?
08F0:    lxi  b,$1600       # Point to Sprite animation sequence table
08F3:    jmp  $0926         # Update Ship Graphic based on position
--------------------------------------------------



08ED
-------------------------------------------------LEFT+RIGHT button&boundary handling?
#  [HL] geeft soort van boundary left/right aan
0900:    ldax $43a0         # v43a0_INput_backup
0903:    cma                #   invert
0904:    ani  $60           # bits #6,5 left right button control actief?
0906:    rz                 # nee

0907:    ani  $40           # bit #6 left button
0909:    jz   $0917         # no

# left movement handling
090C:    mov  a,m           #   v43c2_SpaceShip-horizPixelpos      (spec.obj0.val2)
090D:    cpi  $09           # LB lowest left edge boundary (in pixs)
090F:    rc
#  LB boundary >= 09
0910:    dcr  m             # decrease current left edge pos
0911:    mvi  a,$ff
0913:    stax $4360         # v4360_indication_LRbuttonPushingPushing (:=FF for movement)
0916:    ret
```

```
0909
-----------------------------------------------------right  button pushed
# right movement handling
0917:    mov  a,m          # v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
0918:    cpi  $c0          # RB highest right edge boundary (in pixs)  bottom right edge
091A:    rnc
#  RB boundary < c0
091B:    inr  m            # increase current right edge pos
091C:    mvi  a,$ff
091E:    stax $4360        # v4360_indication_LRbuttonPushingPushing (:=FF for movement)
0921:    ret



(08f3) 0959
------------------------------------------------------ perform Ship or Laser anim sequnce
#spaceship sprite index (B=1600)
#laserbeam (when B=1620)
0926:    mov  a,m          # HL 43c2 (spec.obj0.val2 spaceship X-index)
0927:    ani  $07          # displacement in 3 bits pixelval to the right
0929:    add  c            # 00 0f 20  (B=1600/1620)
092A:    mov  c,a
092B:    ldax b            # vb b=1600 [via PC=08f0] of b=1620 via [0956]
092C:    dcr  l
092D:    mov  m,a          # 43c1 spec.obj0.val1 (2x2 sprite offset @1400)
092E:    ret

" animation sequence for pulsating space ship
1600:  [10 14 18 1C 00 04 08 0C] 20 22 24 26 28 2A 2C 2E  first block is SPACE SHIP, second smallbirds

  animation sequence for the laser projectile, within 1 per char laser shifts 1 pixel to the right
1620:  [50 51 52 53 54 55 56 57] FF FF FF FF FF FF FF FF   laser bullets 8x pixel shifted
"

08A6, 08B4
------------------------------------------------------ normal/highspeed laser fire positioning
#[DE] set by xchg (0936):
# HL := $43c4 (08A3) regular laser projectile
# HL := $43c8 (08B1) rapid laser projectile
0930:    mov  a,m              # 43c4 special obj slot #1 (laser fire)
0931:    ani  $08              #
0933:    jnz  $0964            #   bit #3 of SHOW state set then skip
0936:    xchg
0937:    mvi  b,$10                # bit#4 FIRE (0001.0000) button to detect
0939:       call $00bb         #Check if FIRE input is active, has INput var bit [B]  value changed?
093C:    rz

093D:    mov  a,m
093E:    ani  $ef              #1110.1111, clearing out bit #4  HIDE state
0940:    mov  m,a
0941:    ldax d                #
0942:    ori  $08
0944:    stax d                    # bit #3 SHOW van spec.obj1.val0 zetten (state lasershot)
0945:    inx  d
0946:    inx  d
0947:    ldax $43c2                # v43c2_SpaceShip-horizPixelpos  (spec.obj0.val2) X-coord
094A:    adi  $04          # bits#76543=col  #210=   adding #100, for right-shifting projectile 4 pixels
094C:    stax d
094D:    inx  d
094E:    ldax $43c3                # v43c3_SpaceShip-vertPixelpos  (spec.obj0.val3) Y-coord
0951:    sui  $08          # for non spaceship affects the row pos up, but it acts here as countdown!
0953:    stax d
0954:    dcx  d
0955:    xchg
0956:    lxi  b,$1620          # 8x spaceship laser sprites each with more pixel displacement
0959:       call $0926         # Update Laser anim sequence
095C:    mvi  a,$30
095E:    stax $4361            # 4361 := 30  v4361_laser-countdown
0961:    ret
```

```
0933
------------------------------------------------------
0964:    inr  l                # 43c4
0965:    inr  l                # 43c5
0966:    inr  l                # 43c6
0967:    mov  a,m              # 43c7 (rapid laser fire) specobj#1 vertical pixel pos (per 8 bits)
0968:    sui  $08              # bits#76543 projectile one row up
096A:    mov  m,a
096B:    cpi  $1f              # bullet reached top position?
096D:    rnc                   # return if not

(0c97, 0cac)
096E:    dcr  l
096F:    dcr  l
0970:    dcr  l
0971:    mov  a,m              # 43c4  spec.obj1 (laser fire)
0972:    ani  $f7              #   clear out bit #3
0974:    mov  m,a
0975:    ret


    # 0978: 7E         mov  a,m
    # 0979: E6 3A      ani  $3a


0882
------------------------------------------------------determine new hi/low spaceship horizontal-pixel boundaries
097a:        ldax $43c2            # v43c2_SpaceShip-horizPixelpos   (spec.obj0.val2)
097d:        mov b,a
097E:    ani  $07
0980:    rlc
0981:    lxi  h,$0b38          # ship left/right boundary data location
0984:    add  l                    # lo offset 38 + horpixelpos*2 [0-F]
0985:    mov  l,a
0986:    mov  a,b
0987:    sub  m
0988:    stax $439e            # v439e_wing.YY_hiboundary
098B:    inx  h
098C:    mov  a,b
098D:    add  m
098E:    stax $439f            # v439f_wing.YY_lowboundary
0991:    ret

    Hexdata "  (0     2     4     6      8     a     c     e   ) v439e_wing.YY_hiboundary  index (sub)
               (   1     3     5     7      9     b     d     f) v439f_wing.YY_lowboundary index (add)
        0B38:  00 08 01 09 02 0A 03 0B  03 0B 02 0A 01 09 00 08
    "


    # ?
    # 0992: 32 9F 43   stax $439f v439f_wing.YY_lowboundary
    # 0995: C9         ret



0523, 087f
------------------------------------------------------new positions for spaceship and lasers
# 3 objects (43c0/4/8) 43c0->spaceship, 43c4->laser, 43c8->laser-extra
09A0:    lxi  b,$43c2          # v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
09A3:    lxi  d,$43e2          # ship pos collision point (upeper left)
(09b2)
09A6:       call $09ba         # bepalen nieuwe wingschermpos
09A9:    inx  b
09AA:    inx  b
09AB:    inx  b
09AC:    inx  d
09AD:    inx  d
09AE:    inx  d
09AF:    mov  a,c
09B0:    cpi  $ce          #c2/c6/ca
```

```
09B2:    jnz  $09a6
09B5:    ret



09A6, 0A87, 0c71
---------------------------------------- determine new screenpointer based on object wingdata xy coord
#09a6, source for: (BC=43c2, DE=43e2) spaceship, laser and extra laser
#0a87, source for: (BC=4b72, DE=4bb2) enemy wings
#0c71, source for: (BC=43ce, DE=43ee) ememy bombs #1, #2, #3 and big bird bombs #1, #2
# B(x,y) -> D-16bit Wing-screenpos pointer,  val := pointer(0a00+hi(#7-#3)), Y:= pointer +1 +lo(#7-#3)
#                                          [ww xx YY ZZ]  X-coord =  #yyyyy000,  Y-coord = #zzzzz000
#   [DE] pointer example here = 4bb2, the YY value of [ww xx YY ZZ]
#   [BC] pointer example here = 4b72, pointing to newpos
09BA:    lxi  h,$0a00       # the 26 pointers of the screencolumns
09BD:    ldax b             #  convert X coordinate to a proper column index (ex B=4b72->65 0110.1001)
09BE:    ani  $f8           #       65->60    1111.1000
09C0:    rrc                #       60->30    0xxx.xx00
09C1:    rrc                #       30->18    00xx.xxx0  next column 0-1f *2 16bits-index
09C2:    add  l             #       l=00->18
09C3:    mov  l,a           #   HL:=0a18 points to 41A0 (15th column)
09C4:    mov  a,m           #   fetch hi-byte of pointer of 41A0
09C5:    stax d             # 4bb2:=41 wingscreenpos highval calculated
09C6:    inx  b             #   same conversion for the Y value  4b73   (ex 4b73=48, 43c3=..)
09C7:    inx  d             #       4bb3
09C8:    inx  h             #       0a19
09C9:    ldax b             #       48
09CA:    ani  $f8           #       48        1111.1000
09CC:    rrc                #       24        0xxx.xx00
09CD:    rrc                #       12        00xx.xxx0
09CE:    rrc                #       09        000x.xxxx   xxxxx is next row 0-1a index
09CF:    add  m             #   0a19->A0 + 09
09D0:    stax d             #  4bb3:=A9 wingschermpos lowval example
09D1:    ret


"0A00 (26 columns pointers)
  screenpositions of columns, 4320 is at the upperleft, 4000 locates upperright  (screen fore-ground mode)
     index    0    2    4    6    8    a    c    e  ... index = col *2, per col 2 bytes for HL videomem pointer
     0A00:  4320 4300 42E0 42C0 42A0 4280 4260 4240  (columns 0-7)
     0A10:  4220 4200 41E0 41C0 41A0 4180 4160 4140  (columns 8-15)
     0A20:  4120 4100 40E0 40C0 40A0 4080 4060 4040  (columns 16-23)
     0A30:  4020 4000                                (columns 24-25)
                      00 00 00 00 00 00 00 00 00 00 00 00
    (0A40:  AA BA AB BB 80 90 81 91 74 7C 75 7D FF FF FF FF)
"

0845, 2150, 2190
----------------------------------------------------------plot enemy (smallbird) fleet?
0A50:    lxi  b,$4b70          # v4b70_wingformation-base      base
0A53:    lxi  d,$4bb0          # v4bb0_wingformation_positions base
(0a64)                         #
0A56:    push b                #
0A57:       call $0718         # SHOW/HIDE enemy/ship?
0A5A:    pop  b                #
0A5B:    mov  a,c              #
0A5C:    adi  $04              # volgende tupple   wingdata
0A5E:    mov  c,a              #
0A5F:    adi  $40              # volgende tupple wingpos (tov wingdata)
0A61:    mov  e,a
0A62:    mov  d,b
0A63:    ana  a                #   z=1 when A==0
0A64:    jnz  $0a56            # do all tupples (until Z==0)
0A67:    ret



0529, 2186, 21ab
----------------------------------------------------Modify wingformations
#  [BC] -> enemy unit wingformation-data [ww xx yy zz],
#  [DE] -> wingdata.pos (NNOO screenpointer, PPQQ screenpointer)
```

```
#  action HIDE/SHOWMODE -> backup PPQQ (= actualpos) to  NNOO (= oldpos)
#  newpos is based on bytes YY and ZZ
0A6C:    lxi  b,$4b70          #   v4b70_wingformation-base [ww xx yy zz]
0A6F:    lxi  d,$4bb3          #   wingdata.pos (QQ in : NNOO PPQQ) ,
(0a96 outer loop)              #
0A72:    push b                #
0A73:    push d                #
0A74:     ldax b               #    wingdata.state  (4b70) Wxyz
0A75:     ani  $18             #   0001.1000  HIDE en SHOW mode
0A77:     jz   $0a8a           #  next object, if both bits are unset
# one of HIDE or SHOW set
# [DE] := PPQQ
# NNOO := PPQQ   (backup of screenposition)
0A7A:     xchg                 #                  DE (4bb3) <-> HL (....)
0A7B:     mov  d,m             #                  D := QQ
0A7C:     dcx  h               #                  4bb2
0A7D:     mov  e,m             #                  E := PP
0A7E:     dcx  h               #                  4bb1        4bb0, 4bb1
0A7F:     mov  m,d             #                  OO := QQ    newpos low-byte
0A80:     dcx  h               #                  4bb0
0A81:     mov  m,e             #  NNOP := PPQQ     NN := PP    newpos hi-byte
0A82:     xchg                 #       HL switch for upcoming call
0A83:     inx  d               #                                          (ex. 4bb1)
0A84:     inx  d               #   pointing to PPQQ screenpointer         (ex. 4bb2)
0A85:     inx  b               #                                          (ex. 4b71)
0A86:     inx  b               #   [ww xx YY zz] points to y-axis value   (ex. 4b72)
0A87:       call $09ba         # bepalen nieuwe wingschermpos
(0a77 next obj)
0A8A:    pop  d                #   4b70
0A8B:    pop  b                #   4bb3
0A8C:    mov  a,c              #
0A8D:    adi  $04              #   volgende tupple
0A8F:    mov  c,a              #
0A90:    mov  a,e              #
0A91:    adi  $04              #   volgende tupple
0A93:    mov  e,a              #
0A94:    cpi  $03              #
0A96:    jnz  $0a72            #   next run
0A99:    ret



08ca
------------------------------ handle barrier
0AA0:    mvi  l,$e2            #   43e2? spaceship screenpos?
0AA2:    mov  d,m              #
0AA3:    inx  h                #
0AA4:    mov  e,m              #
0AA5:       call $0210         #   cursor met 1 kolom positie naar links
0AA8:    dcx  d                #
0AA9:    lxi  b,$0404          #
0AAC:    mvi  l,$a6            #   v43a6_barrier_repeat_timer
0AAE:    dcr  m
0AAF:    mov  a,m
0AB0:    lxi  h,$17f0          # #romdata endmarker,  explosion/barrier spaceship?
0AB3:    cpi  $c0
0AB5:    jz   $0b48

0AB8:    lxi  h,$1770          #romdata startmarker?
0ABB:    ani  $0c
0ABD:    rlc
0ABE:    rlc
0ABF:    add  l
0AC0:    mov  l,a
0AC1:    jmp  $0ad6            # transfer block[B,C] of romdata to screen

Hexdata ''
    0AC0  6F C3 D6 0A FF FF FF FF FF FF FF FF FF FF FF FF   oÃÖ.ÿÿÿÿÿÿÿÿÿÿÿÿ
    0AD0  FF FF FF FF FF FF D5 C5 7E 12 23 13 05 C2 D8 0A   ÿÿÿÿÿÿÕÅ~.#..ÂØ.
    ''
```

```
(0ac1,0ae6),0B48,(210d,2337,2473)
------------------------------------------ transfer block[B,C] of romdata(HL) to screen(DE)
(0ae6 C-outerloop)
0AD6:     push d
0AD7:     push b
(0add B-innerloop)
0AD8:     mov  a,m
0AD9:     stax d
0ADA:     inx  h
0ADB:     inx  d
0ADC:     dcr  b
0ADD:     jnz  $0ad8
0AE0:    pop  b
0AE1:    pop  d
0AE2:       call $0217          # cursor met 1 kolom positie naar rechts
0AE5:    dcr  c
0AE6:    jnz  $0ad6
0AE9:    ret



JT1:4->040d
-------------------------------------------------- screen scrolling (starfield)?
0AEA:    lxi  h,$43b9           # v43b9_videoscroll
0AED:    mov  a,m               #
0AEE:    ani  $f8               #
0AF0:    mov  m,a               #
0AF1:    stax $5800              VIDEO Scroll Reg
0AF4:    mvi  l,$e2             # 43e2
0AF6:    mov  d,m               #
0AF7:    inr  l                 #
0AF8:    mov  e,m               #
0AF9:       call $0210          # cursor met 1 kolom positie naar links
0AFC:    dcx  d                 #
0AFD:    nop                    #
0AFE:    mvi  l,$a5             # v43a5_JT1counter
0B00:    dcr  m                 #
0B01:    mov  a,m               #
0B02:    jz   $0b15             #
0B05:    cpi  $20               #
0B07:    jc   $0ba0             #
0B0A:    jz   $0380             # schoon foreground behalve HUD (1e 3 regels)
0B0D:    jmp  $0bba


(0b02)
0B15:    dcr  l
0B16:    mvi  m,$05
0B18:    dcr  l
0B19:    mov  a,m
0B1A:    adi  $90
0B1C:    mov  l,a
0B1D:    mov  a,m
0B1E:    ana  a
0B1F:    rz

0B20:    dcr  m
0B21:    push h
0B22:       call $0367          # werk nr. of lives bij
0B25:    pop  h
0B26:    mov  a,m
0B27:    ana  a
0B28:    rz

0B29:    mvi  l,$a4             # v43a4_JT1-indexer
0B2B:    mvi  m,$00             #                   := 0
0B2D:    ret


    Hexdata "
    0B30:  FF F0 E0 B0 C0 D0 C0 B0 00 08 01 09 02 0A 03 0B
```

```
    0B40:  03 0B 02 0A 01 09 00 08
    "

0ab5
------------------------------------------------------ ship got hit (explosion) handling?
# [WW XX YY ZZ]
0B48:       call $0ad6      # transfer block[B,C] of romdata(HL) to screen(DE)
0B4B:    lxi  h,$43c0       # special object slot #0 (the space ship)
0B4E:    mvi  m,$0c         # WW := 0000.1100   (preparing to SHOW JT3:4 index 07d2, do ship animation)
0B50:    inr  l             #  43c1
0B51:    mvi  m,$0c         # XX:= 0c (space ship posture to go for)
0B53:    inr  l             #  43c2
0B54:    mov  a,m           # fetching YY part, the current row of the space ship
0B55:    ani  $f8           # xxxx.x000
0B57:    ori  $03           # xxxx.x011 setting bits #10
0B59:    mov  m,a           # fill YY  (column of screen index)
0B5A:    ret



(03bf), JT1:5->040d
-----------------------------------------------------game/match over handling
#   reset v43989_16bits_counter
0B60:    lxi  h,$43a5           # v43a5_JT1counter
0B63:    inr  m                 # verhogen
0B64:    mov  a,m
0B65:    cpi  $40
0B67:    jz   $03a0             # skip bij $40 bereikt -> schoon background met 00

0B6A:    lxi  h,$1a00           # (1A00:  4328)    schermpos  copyright/game over
0B6D:    mvi  c,$01
0B6F:    cpi  $80
0B71:    jnz  $0b95             # <>$80 dan 1 regel afdrukken?
0B74:    lxi  h,$43a4           # v43a4_JT1-indexer
0B77:    mvi  m,$00             #               := 00
0B79:    mvi  l,$90             # v4390_PL1numlives
0B7B:    mov  a,m
0B7C:    inr  l                 # v4391_PL2numlives
0B7D:    ora  m
0B7E:    rnz                    # numlives van een van de spelers nog niet 0
#  reset v43989_16bits_counter counter and v43a2_curplayer to zero
0B7F:    xra  a                 # eor [A:=0]
0B80:    mvi  l,$98             # v43989_16bits_counter
0B82:    mov  m,a               #                   := 00
0B83:    inr  l                 # v43989_16bits_counter
0B84:    mov  m,a               #                   := 00
0B85:    mvi  l,$a2             # v43a2_curplayer
0B87:    mov  m,a               #     := 00   v43a2_curplayer:=0 bij game-over
0B88:    inr  l                 # v43a3_PLtoggle
0B89:    mov  a,m
0B8A:    ana  a
0B8B:    rz                     # return als v43a3_PLtoggle=0 is
#  geen gameover?
0B8C:    mvi  m,$00             # v43a3_PLtoggle := 00
0B8E:    lxi  b,$0100           # b=01,c=00 (gebruikt voor palette waarden)
0B91:       call $0460          # schakel per karakter over naar ander palette
0B94:    ret



0b71
------------------------------------------------------
0B95:       call $01d0          # PrintLNs
0B98:       call $01e4          # HUD2 afdrukken: (c) 1980 TAITO CORPORATION
0B9B:    jmp  $1df0


0b07
------------------------------------------------------
0BA0:    lxi  h,$43b8       #  v43b8_staging0-F_videobit1   (influences video scroll reg)
0BA3:    mov  a,m
```

```
0BA4:    ani  $0f           # maintain low-end nibble
0BA6:    cpi  $04           # stage 4 [game over]
0BA8:    rc                 # return if  v43b8_staging0-F_videobit1 < 4

0BA9:    cpi  $09           # stage 9 [upcoming boss-ship]
0BAB:    rnc                # return if  v43b8_staging0-F_videobit1 <= 9

#   --- baseship scrolldown anim stage
0BAC:    inr  l             #
0BAD:    xra  a             #       eor with itself, trick to set A to zero
0BAE:    mov  m,a           #  v43b9_videoscroll := 0
0BAF:    stax $5800         VIDEO Scroll Reg  := 0
0BB2:    jmp  $03a0         # schoon background met 00



0b0d
-------------------------------------------------
0BBA:    mov  b,a
0BBB:    rrc
0BBC:    jnc  $0fc0         # bit #0 in A not set

0BBF:    rrc                # set carry
0BC0:    mov  a,b
0BC1:    jc   $2070         # ship explosion animation sequence preperation+execution?
0BC4:    jmp  $20e8



0104
------------------------------------------------- enemies in score average table
0BCA:    lxi  h,$42d0       #   schermpos
0BCD:    lxi  b,$ffdf       #
0BD0:    mvi  m,$64         # =smallbird-leftpart
0BD2:    dad  b             #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
0BD3:    inx  h             #
0BD4:    mvi  m,$65         # =smallbird-rightpart
0BD6:    lxi  h,$42f2       #   schermpos
0BD9:    lxi  d,$0a40       # kleine vogel naar links en rechts vleugels uit
0BDC:       call $3538      #   8 bytes verplaatsen (0a40->42f2 schermpos)
0BDF:    lxi  h,$4b15       #   background schermpos
0BE2:    lxi  d,$3c00       # big bird
0BE5:       call $3528      #   12 bytes verplaatsen (3c00->4b15 background)
0BE8:    lxi  h,$4ad8       # fixed location on background screen (3, 19?)
0BEB:    lxi  d,$0a48       # boss-enemy
0BEE:       call $3548      #   4 bytes verplaatsen (0a48->4ad8 background)
0BF1:    ret

    "<-- 0BD9
    0A40:  [AA BA AB BB] [80 90 81 91]  74 7C 75 7D FF FF FF FF  smallbird left/right flight
    3C00:  [E8 00 E9 00 C4 C6 C5 C7 EA 00 EB 00]  00 00 EC 00    bigbird
    0A40:  AA BA AB BB 80 90 81 91  [74 7C 75 7D]  FF FF FF FF   boss
    "

0e6b
-------------------------------------------------
#  screen before enemy (small)bird gets hit (only) during bigboss fight scene
# HL [4b92 gets 4b60 =[2c]explosion sequences #0-7 @2800-2900-2a00-2b00-2c00-2e00-2f00  ]
0C00:    push h
0C01:     mov  a,l          # 4b92
0C02:     sui  $72          #
0C04:     rrc               #
0C05:     adi  $50          # 4b60
0C07:     mov  l,a
0C08:     mov  a,m          #4bc0=2c
0C09:     inr  l
0C0A:     mov  l,m          #4b61=0c
0C0B:     mov  h,a          #HL=2c0c
0C0C:     lxi  d,$0c04      #0c 04 as DE data
0C0F:     mov  a,m
0C10:    pop  h
```

```
0C11:    cpi  $07
0C13:    jc   $0ea4

0C16:    cpi  $09
0C18:    jnc  $0ea4

0C1B:    lxi  d,$1020
0C1E:    mvi  a,$ff
0C20:    stax $4369         # v4369_BarrierShield-Hit := FF (bullit collision detected)
0C23:    jmp  $0ea4



2163, 2183, 21bf, 3467
------------------------------------------------------
0C40:    lxi  h,$43ff       # SP end-range
0C43:    mvi  b,$05         # 5 words
0C45:       call $088b      # make the word copies
0C48:       call $0c56      # special objects (43cc-43df) loop
0C4B:       call $0c6b
0C4E:       call $0cd8
0C51:    ret



0C48
------------------------------------------------------special enemy objects hit detection loop
# loop until end of SPEC.OBJ  (43cc-43df)  reached

# 43cc/cd/ce/cf = enemy birds bomb #1
# 43d0/d1/d2/d3 =  ''              #2
# 43d4/d5/d6/d7 =  ''              #3
# 43d8/d9/da/db =  big bird bomb #1
# 43dc/dd/de/df =  big bird bomb #2
0C56:    lxi  h,$43cc       #  (start with special object slot #3 enemy bombs)
(0c64)
0C59:    push h
0C5A:       call $0c84      # test and handle bit#3 (hit detection?)
0C5D:    pop  h
0C5E:    mov  a,l
0C5F:    adi  $04           # next object
0C61:    mov  l,a
0C62:    cpi  $e0           #reached 43e0?
0C64:    jnz  $0c59
0C67:    ret



0C4B
------------------------------------------------------obj new screenpositions (?)
  # 5 objects?  base@ 43cc/d0/d4/d8/dc   movement borders of enemy birds?
  # 43e0/1 shippos, 43e2/3 idem (bottomleft)
  # 43e4/5 ship laser projectile (bottom visible)
  # 43e6/7  idem but 1 line above for collision detection purposes / or for the extra hi-speed laser bullet?
  # 43e8/9  1 line above spaceship? (leftpos), 43ea/b idem
  # 43ec/d  leftpos above (last line enemy smallbirds?), 43ee/f idem
  # 43f0/1  bird bomb pos1, 43f2/3 idem
  # 43f4/5  bird bomb pos2 (same bomb as above), 43f6/7 idem
0C6B:    lxi  b,$43ce           # smallbird bombs, bigbird bombs, ...
0C6E:    lxi  d,$43ee           # (?? left pos above last line enemy smallbirds??)
(0c7d)
0C71:       call $09ba      #   bepalen nieuwe wingschermpos
0C74:    inx  b
0C75:    inx  b
0C76:    inx  b
0C77:    inx  d
0C78:    inx  d
0C79:    inx  d
0C7A:    mov  a,c
0C7B:    cpi  $e2               # 43ce/d2/d6/da/de
```

```
0C7D:    jnz  $0c71
0C80:    ret



0C5A
------------------------------------------------------ spec obj hit detection handling
0C84:    mov  a,m
0C85:    ani  $08       #  bit #3
0C87:    rz             #  is not set

### (sprite handling of spaceship during  bird fight)
#bit 3=1
0C88:    nop
0C89:    nop
0C8A:    inr  l             # spec obj 2x2 sprite index??)
0C8B:    mov  a,m
0C8C:    xri  $04           # bit #3 toggle (hit detection)
0C8E:    mov  m,a
0C8F:    inr  l
0C90:    inr  l
0C91:    mov  a,m
0C92:    adi  $04           # next object
0C94:    mov  m,a
0C95:    cpi  $f9           # until 43f9? (within collision objs)
0C97:    jnc  $096e         # clear out #3 current object (<= c8 -> 43c8)

0C9A:    dcr  l
0C9B:       call $0cb4      # pointer reached 43dc/e9 check
0C9E:    mov  d,h
0C9F:    mov  a,l
0CA0:    adi  $20
0CA2:    mov  e,a
0CA3:    xchg
0CA4:    mov  b,m
0CA5:    inx  h
0CA6:    mov  c,m
0CA7:    ldax b
0CA8:    xchg
0CA9:    inr  l
0CAA:    cpi  $e8
0CAC:    jnc  $096e
0CAF:    ret



(0C9B)
------------------------------------------------------
# 43dc - 43e9
0CB4:    cpi  $dc       # reached $43dc?
0CB6:    rc             # x < dc
  # > 43dc
0CB7:    cpi  $e9       # reached $43e9
0CB9:    rnc            # x >= e9
  # x < e9
0CBA:    ldax $439f             # v439f_wing.YY_lowboundary
0CBD:    cmp  m
0CBE:    rc

0CBF:    ldax $439e             # v439e_wing.YY_hiboundary
0CC2:    cmp  m
0CC3:    rnc

"
43D0                                     00 58 00 20   .X. .X. .X. .X.
43E0  41 7B 41 7B 41 A4 41 A3 43 3A                    A{A{A¤A£C:C$C$

"

0F46
------------------------------------------------------
```

```
0CC4:    mvi  a,$04           #   0cc4 Invincibility toggle?
0CC6:    stax $43a4           #    v43a4_JT1-indexer : = 04   screen scrolling?
0CC9:    mvi  a,$60           #
0CCB:    stax $43a5           #    v43a5_JT1counter := $60
0CCE:    mvi  a,$10           #
0CD0:    stax $4363           #    v4363_Spaceship-hit := $10  (SpaceShip hit detected)
0CD3:    ret



0C4E
-----------------------------------------------------
0CD8:    lxi  b,$43cc          # (special object slot #3, smallbird enemy bomb #1 )
0CDB:    lxi  d,$43ec          # (left pos above, last line enemy smallbirds?)
(0CEC)
0CDE:    push b
0CDF:       call $0718         # SHOW/HIDE enemy/ship?
0CE2:    pop  b
0CE3:    mov  a,c
0CE4:    adi  $04              # 43d0/43e4  smallbird bomb #2
0CE6:    mov  c,a
0CE7:    adi  $20              # 43f0/43f4  smallbird bomb #3
0CE9:    mov  e,a
0CEA:    mov  d,b
0CEB:    ana  a
0CEC:    jnz  $0cde
0CEF:    ret



(0F60)
-----------------------------------------------------quit prematurely (Z=0 bitflag?)
#  we have detected spaceship-smallbird object collision
0CF4:    pop  d
0CF5:    pop  b
0CF6:    ret

Hexdata ? "
    0CF0  FF FF FF FF D1 C1 C9 FF FF FF FF FF FF FF FF FF   ÿÿÿÿÑÁÉÿÿÿÿÿÿÿÿÿ
    0D00  FF FF FF FF FF FF FF FF 21 93 43 34 7E E6 07 C0   ÿÿÿÿÿÿÿÿ!.C4~æ.À
    0D10  2C 2C 7E 3C E6 0F 77 C9 FF FF FF FF 01 70 4B 21   ,,~<æ.wÉÿÿÿÿ.pK!
    "
???
-----------------------------------------------------
   0D08:    lxi  h,$4393        #   v4393_JT6_indexer
   0D0B:    inr  m
   0D0C:    mov  a,m
   0D0D:    ani  $07
   0D0F:    rnz

   0D10:    inr  l
   0D11:    inr  l
   0D12:    mov  a,m
   0D13:    inr  a
   0D14:    ani  $0f            # maintain low-end nibble
   0D16:    mov  m,a            # 0-15 value counter
   0D17:    ret



2166, 21a5
-----------------------------------------------------
0D1C:    lxi  b,$4b70        #   v4b70_wingformation-base
0D1F:    lxi  h,$4b50        #   v4b50_smallbird-romlists
(0d2b)
0D22:       call $0d30       # fleet position stepping?
0D25:    inr  c
0D26:    inr  c
0D27:    inr  l
0D28:    mvi  a,$b0
0D2A:    cmp  c              # B reaches 4bb0?
```

```
0D2B:    jnz  $0d22         # no, continue with next pair
0D2E:    ret



0D22
------------------------------------------------------ fleet position stepping [wxYZ]
0D30:    mov  d,m
0D31:    inx  h
0D32:    ldax b          #4b70 wingdata
0D33:    inx  b
0D34:    inx  b          # 4b72 [wxYz]
0D35:    ani  $08        #bit #3
0D37:    rz              #return not set

#ex. smallbird from demo stage #1 dives 4b58 [13 54], 1354 = 1c
0D38:    mov  e,m
0D39:    xchg
0D3A:    mov  a,m
0D3B:    rlc             # *2   ex 1c*2=38
0D3C:    adi  $00
0D3E:    mov  l,a
0D3F:    mvi  h,$17      # 1700+x   ex. 1700+38
0D41:    xra  a          # [A:=0]
0D42:    cmp  m
0D43:    jz   $0d4f

0D46:    inx  h          # 1700+x+1
0D47:    cmp  m              # == 0?
0D48:    jz   $0d5e          # if so

0D4B:    dcx  h
0D4C:    ldax b              # ex. 1738 -> 4c
0D4D:    add  m              # + -4 (=FC)
0D4E:    stax b          # [wxYz]
(0d43)
0D4F:    inx  b
0D50:    inx  h
0D51:    ldax b
0D52:    add  m
0D53:    stax b          #?? f.e. 4b73,4b83  wwxxyyZZ part
0D54:    dcx  b
0D55:    ani  $07
0D57:    xchg
0D58:    rnz

0D59:    inr  m              # [13 55]
0D5A:    ret

"          0  1   2  3   4   5  6  7
    1700:  FF FF [01 00] FF [00 04 00] [FC 00] 00 [FC] 00 [04 04 FE
    1710:  FC FE 04 02 FC 02] 00 [04] 00 [04] 00 [04] 00 [04 FF FF
    1720:  FC 00 FC 00 FC 00 FC 00 04 00 04 00 04 00 04 00
    1730:  04 FC 04 04 FC 04 FC FC FC FC FC 04 04 04 04 FC      #1738 -> FC (two complement: FF=-1, FC=-4 )
    1740:  08 00

"



0d48
------------------------------------------------
#
0D5E:    dcx  h
0D5F:    ldax b
0D60:    add  m
0D61:    stax b          #f.e. 4b72, 4b76, ...  wwxxYYzz part
0D62:    ani  $07
0D64:    xchg
0D65:    rnz

0D66:    inr  m          # odd soundval increase (4b51/53/55/57/59/5b)
```

```
0D67:    ret



2170,21a8
------------------------------------------------------
0D70:    lxi  b,$4b70          #   v4b70_wingformation-base
0D73:    lxi  h,$4b50          #   v4b50_smallbird-romlists
(0d80)
0D76:       call $0d86
0D79:    mov  a,c
0D7A:    adi  $04
0D7C:    mov  c,a
0D7D:    mvi  a,$b0
0D7F:    cmp  c                #  until 4bb0
0D80:    jnz  $0d76
0D83:    ret


0D76
------------------------------------------------------
###starfield background plot when not occupied by wing objects? <-- no
0D86:    mov  d,m
0D87:    inx  h
0D88:    mov  e,m
0D89:    inx  h             # fetch DE (from mem)
0D8A:    ldax b             # 4b70 wingdata
0D8B:    ani  $08
0D8D:    rz

0D8E:    xchg               # DE<->HL
0D8F:    mov  a,m
0D90:    ana  a
0D91:       cz   $0dde      # jump if end of list reached [A=0]

0D94:    mov  l,a
0D95:    rlc
0D96:    add  l
0D97:    adi  $a0
0D99:    mov  l,a
0D9A:    mvi  h,$16         # 16a0+3+val (base 16a3 + ...)
0D9C:    ldax b             # WW part
0D9D:    ani  $f8
0D9F:    ora  m             # (A,b,c)   value from HL=1000 / ..., ex. 01
0DA0:    stax b             #f.e. 4ba0 (WW value of smallbird to 39) WWxxyyzz part
0DA1:    inx  b             # XX paet
0DA2:    inx  b             # YY part
0DA3:    inx  b             # ZZ part
0DA4:    inx  h             # ex. 1001 = 02
0DA5:    mov  a,m           #  (a,B,c)
0DA6:    inx  h             # ex. 1001
0DA7:    rrc
0DA8:    jc   $0dbb     # jump when bit #0 was set

0DAB:    rrc
0DAC:    jc   $0dcc     # jump when bit #1 was set

0DAF:    ldax b
0DB0:    rrc
0DB1:    ani  $03
0DB3:    add  m
0DB4:    dcx  b
0DB5:    jmp  $0dd2

"
1000:  [01 01 01 01 02 02 02 02 02 02 02 02 01 01 01 01
1010:  00]

@16a3        (a  b  c) triplets  for [WXyz] modification
  (01 02 08) (01 02 08) (01 02 0C) (01 02 10) (03 04 14)
  (03 04 18) (04 01 88) (04 01 90) (04 01 80) (04 01 80)
```

```
  (03 04 70) (03 04 74) (03 04 78) (03 04 7C)

    1608:   [20 22 24 26 28 2A 2C 2E]
"

(0da8)
-----------------------------------------------------
0DBB:    ldax b
0DBC:    rrc
0DBD:    ani  $03
0DBF:    add  m             #(a,B,c)
0DC0:    mov  h,a
0DC1:    dcx  b
0DC2:    ldax b
0DC3:    ani  $04
0DC5:    add  h
0DC6:    jmp  $0dd2


(0dac)
-----------------------------------------------------
0DCC:    dcx  b             #YY part
0DCD:    ldax b
0DCE:    rrc
0DCF:    ani  $03
0DD1:    add  m             #(a,b,C)
(0dc6)
0DD2:    mov  l,a
0DD3:    mvi  h,$16         #16xx  starfield background
0DD5:    mov  a,m           #ex 160b
0DD6:    dcx  b
0DD7:    stax b             #f.e. 4b81  wwXXxxyyzz part
0DD8:    dcx  b
0DD9:    xchg
0DDA:    ret


0d91
-----------------------------------------------------tranfer to romlistptr from preset
# (4394,4395) index for 1000 (ENEMY SPRITE MOVEMENT offsets)
# example DE=4b62 (4b66, 4b6a, 4b6c ..
0DDE:    dcx  d             # DE--
0DDF:    dcx  d             # DE--
0DE0:    ldax $4394         #   v4394_4b50copy offset
0DE3:    stax d             # 4b50 + evenIndex  ( 10)     4b50 := 4394
0DE4:    mov  h,a           #
0DE5:    inx  d             # DE++
0DE6:    ldax $4395         # 4395 (var used at JT6 related code)
0DE9:    stax d             #   4b51 = 4395
0DEA:    mov  l,a           #
0DEB:    inx  d
0DEC:    mov  a,m
0DED:    ret


2003
-----------------------------------------------------
0DF0:    lxi  b,$43c4       #  (special object slot #1 laser regular speed beam)
0DF3:    lxi  h,$43e6       # spaceship laser projectile (1 ahead pos)
0DF6:       call $0e10

JT5: (<-0EE5) #### when SMALLBIRD HIT
-----------------------------------------------------
0DF9:    lxi  b,$43c8       #  (special object slot #2 laser rapid speed beam)
0DFC:    lxi  h,$43ea       # one line above spaceship (left pos screen)
0DFF:    jmp  $0e10

-------------------------------
0E02:    lxi  b,$43cc       # (special object slot #3 smallbird enemy bomb #1)
0E05:    lxi  h,$43ee       # left pos line above enemy smallbirds
0E08:       call $0e10
```

```
0E0B:    ret



0DF6, 0E08, (0dff)
------------------------------------------------------ laser hit enemy detection?
0E10:    ldax b             # [B=43c4/43cc/43c8]
0E11:    ani  $08
0E13:    rz                 # return if bit #3 SHOW obj not set

0E14:    mov  d,m           #   DE := [43e6/43ee/43ea] collision positions
0E15:    inr  l
0E16:    mov  e,m
0E17:    ldax d
0E18:    cpi  $c0           # did the laser hit an object above and including char c0?
0E1A:    rnc                # yes, skip     *[43e6/43ee/43ea] >= $c0 ?     1100.0000
                            # or
0E1B:    cpi  $60           #  below char 60 ?
0E1D:    rc                 #   also skip   *[43e6/43ee/43ea] < $60 ?        0110.0000

0E1E:    cpi  $68           #  flighing small bird signature
0E20:    jnc  $0e39         #       *[43e6/43ee/43ea] >= $68 ?   0110.1000
#laser hit char [60-68] smallbird type
0E23:    ani  $07           #       0000.0xxx   sb
0E25:    rlc                #       0000.xxx0
0E26:    rlc                #       000x.xx00
0E27:    adi  $40           #       010x.xx00
0E29:    mov  l,a           #
0E2A:    mvi  h,$17         # [HL := 8 stapwaarden tussen $1740 -- $175c] 1740 + sb *4
0E2C:    inx  b             #
0E2D:    inx  b             #
0E2E:    ldax b             # [B=43c6/43ca/43ce]  [ww xx YY zz]
0E2F:    ani  $07           # YY bits #2,1,0  [0-7]
0E31:    cmp  m             #      compare value-YYbits to 08/01/08/04/08/08/08/08
0E32:    rnc                #                               c/ ?/ c/ ?/ c/ c/ c/ c

"
    17xx    40=char60     44=char61     48=char62     4c=char63
    1740:  [08] 00 00 FF [01] 00 F8 FF [08] 01 02 FF [04] 00 FA FF
    1750:  [08] 01 04 FF [08] 00 FC FF [08] 05 06 FF [08] 00 FE FF
    17xx    50=char64     54=char65     58=char66     5c=char67
"

0E33:    inx  h             # base 1740+1
0E34:    cmp  m             #      compare value-YYbits to 00/00/01/00/01/00/05/00
0E35:    rc                 #

0E36:    jmp  $0e70

"
    17xx       41            45            49            4d
    1740:  08 [00] 00 FF 01 [00] F8 FF 08 [01] 02 FF 04 [00] FA FF
    1750:  08 [01] 04 FF 08 [00] FC FF 08 [05] 06 FF 08 [00] FE FF
               51            55            59            5d
"


0e20
------------------------------------------------------
0E39:    inx  b
0E3A:    inx  b
0E3B:    ldax b
0E3C:    mov  d,a
0E3D:    inx  b
0E3E:    ldax b
0E3F:    ani  $f8
0E41:    mov  e,a
0E42:    lxi  h,$4b70          #   v4b70_wingformation-base
(0e52)
0E45:    mov  a,m
0E46:    inx  h
```

```
0E47:    inx  h
0E48:    ani  $08              #  bit #3
0E4A:       cnz  $0e58         #   is set, call

0E4D:    inx  h
0E4E:    inx  h
0E4F:    mvi  a,$b0
0E51:    cmp  l
0E52:    jnz  $0e45            # next tupple until 4bb0 reached
0E55:    ret


oe4a::
-------------------------------------------------
#  (bigboss enemy smallbird hit staging)
0E58:    mov  a,d
0E59:    cmp  m
0E5A:    rc

0E5B:    mov  a,m
0E5C:    adi  $08
0E5E:    cmp  d
0E5F:    rc

0E60:    inx  h
0E61:    mov  a,m
0E62:    dcx  h
0E63:    adi  $04
0E65:    cmp  e
0E66:    rc

0E67:    sui  $0c
0E69:    cmp  e
0E6A:    rnc

0E6B:    jmp  $0c00




 "
    17xx          42            46            4a            4e
    1740:  08 00 [00] FF 01 00 [F8] FF 08 01 [02] FF 04 00 [FA] FF
    1750:  08 01 [04] FF 08 00 [FC] FF 08 05 [06] FF 08 00 [FE] FF
    17xx          52            56            5a            5e
 "


0e36
-------------------------------------------------
# vb BC -> 43c6, HL = 1741          # set DE
0E70:    inx  h
0E71:    ldax b             #           43c4+2
0E72:    ani  $f8
0E74:    add  m
0E75:    mov  d,a
0E76:    inx  b
0E77:    ldax b
0E78:    ani  $f8
0E7A:    mov  e,a

0E7B:    lxi  h,$4b70       #   v4b70_wingformation-base
(0e8b)
0E7E:    mov  a,m
0E7F:    inx  h
0E80:    inx  h
0E81:    ani  $08
0E83:       cnz  $0e90      #   jump if wingdata $4b70/74/78/7c/80/../ac   bit #3 set

0E86:    inx  h
0E87:    inx  h
```

```
0E88:    mvi  a,$b0
0E8A:    cmp  l
0E8B:    jnz  $0e7e         # go until 4bb0
0E8E:    ret



0e83::
--------------------------------
0E90:    mov  a,m
0E91:    adi  $02
0E93:    cmp  d
0E94:    rc

0E95:    sui  $05
0E97:    cmp  d
0E98:    rnc

0E99:    inx  h
0E9A:    mov  a,m
0E9B:    dcx  h
0E9C:    ani  $f8
0E9E:    cmp  e
0E9F:    rnz


0EA0:    lxi  d,$0c02       # set DE
0EA3:    nop
0EA4:    dcx  h
0EA5:    dcx  h
0EA6:    dcx  b
0EA7:    dcx  b
0EA8:    dcx  b
0EA9:    ldax b             #fe 43c4 (FGscreen)
0EAA:    ani  $f7
0EAC:    stax b             #bit3 clear

0F4E
------------------------
0EAD:    mov  a,m
0EAE:    ani  $f7
0EB0:    mov  m,a           #bit3 clear

0EB1:    mov  a,l
0EB2:    adi  $42
0EB4:    mov  l,a           # HL := +42

0EB5:    mov  b,m
0EB6:    inx  h
0EB7:    mov  c,m


    " DEAD-ANIM slots (16 bytes, per slot 4 bytes) slots for bird death animations.
                        flag?   BCD?    ?       ?
    Slot Number Type    Byte 1  Byte 2  Byte 3  Byte 4
    1   Regular         4370    4371    4372    4373
    2   Regular         4374    4375    4376    4377
    3   Special         4378    4379    437A    437B
    4   Special         437C    437D    437E    437F
    "

0EB8:    lxi  h,$4378       ##  4378 [slot for special animation]
0EBB:    mov  a,d
0EBC:    cpi  $10           ##   Are we to use the special animation?
0EBE:    jz   $0ec3        ##   Yes, skip next step


0EC1:    mvi  l,$70         ##  4370  (4378<>10) Else [slot for regular animation]
(0ebe)
0EC3:    mov  a,m           ##   Load timer from this slot
0EC4:    ana  a             ##   Is this slot available?
```

```
0EC5:    jz   $0ed5         ##   Yes, skip ahead, we will use this slot

0EC8:    inr  l
0EC9:    inr  l
0ECA:    inr  l
0ECB:    inr  l             ##  4374  Increase HL by 4. [now at 2nd or 4th slot]
0ECC:    mov  a,m           ##   Load timer from this slot
0ECD:    ana  a             ##   Is this slot available?
0ECE:    jz   $0ed5         ##   yes, skip ahead, we will use this slot

## else use the next slot.  Bugged when birds are flying upwards.
## source of 204K bug. HL becomes #4380 which is start of score.

0ED1:     inr  l
0ED2:     inr  l
0ED3:     inr  l
0ED4:     inr  l                ##  4378   Increase HL by 4 [now at 3rd or 5th slot]


    #   HISCORE SAVE BUG FROM DE and B
    # 0ED5: Store D into byte 1
    # 0ED6: Next byte
    # 0ED7: Store E into byte 2. score becomes 20xxxx
    # 0ED8: Next byte
    # 0ED9: Store B into byte 3. score becomes 2041xx or 2042xx

(0ec5,0ece)
0ED5:     mov  m,d          #               normally 4370/4374/4378?, bug 4380?
0ED6:     inr  l             #
0ED7:     mov  m,e           #
0ED8:     inr  l             #
0ED9:     mov  m,b          #   [BC = collision location vb 419c in demo]
0EDA:     inr  l                #   voorbeeld 4370+2 wordt de videoram pos van de collision
0EDB:     mov  m,c          #

-------------------------------------------------------- JUMP TABLE 5
0EDC:     mvi  l,$64             #   v4364_enemy-hit-detected  (FF=true)
0EDE:     mvi  m,$ff             #       := FF
0EE0:     mvi  l,$ba             #   v43ba_smallbird_fleetsize number of enemy birds active ?
0EE2:     dcr  m                 #       -= 1

0EE3:     pop  h                 # fetch previous call return  (no use)
0EE4:     pop  h                 # fetch call return           (   use)
0EE5:     pchl                   JT5 JUMP!  # (0027 SHIP HIT) (0DF9 SMALLBIRD HIT) (0049 SHIP HIT game-over)
------------------------



Hexdata "
    0EE6                   FF FF FF FF FF FF FF FF FF FF   .º5ááéÿÿÿÿÿÿÿÿÿÿ
    0EF0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
    "

2196
-------------------------------------------------------- barrier enemy collision?
0F00:    lxi  h,$43a6       # v43a6_barrier_repeat_timer SHIELD   ??DD=NO_SHIELD, 00=READY,
0F03:    mov  a,m           #
0F04:    cpi  $c0           #
0F06:    jnc  $0f74         # shield is on? (>= $c0)

0F09:    mvi  l,$e2         # 43e2
0F0B:    mov  d,m
0F0C:    inr  l
0F0D:    mov  e,m           # DE <-(43e2:43e3) contains topleft screenpos of spaceship
0F0E:    lxi  b,$0202       # B= #rows , C= #coloms
0F11:       call $0f56      # collision detection? (and movement of spaceship?)
0F14:    rz                 # no

0F15:    nop                #yes, there was a collision
0F16:    nop
0F17:    lxi  h,$439e       # v439e_wing.YY_hiboundary
```

```
0F1A:    mov  a,m
0F1B:    sui  $06
0F1D:    mov  b,a
0F1E:    inr  l
0F1F:    mov  c,m
0F20:    lxi  h,$4b70       #       v4b70_wingformation-base[WXYZ] enemy#0
(0f30)
0F23:    mov  a,m           #
0F24:    inr  l
0F25:    inr  l
0F26:    ani  $08           #       bit #3 van W
0F28:       cnz  $0f38
0F2B:    inr  l
0F2C:    inr  l             #       .. next tupple
0F2D:    mvi  a,$b0
0F2F:    cmp  l
0F30:    jnz  $0f23         # hL tot aan 4bb0
0F33:    ret


0f28::
-------------------------------------------------
#value range (d2,e7)
0F38:    inr  l
0F39:    mov  a,m
0F3A:    dcr  l
0F3B:    cpi  $d2
0F3D:    rc

0F3E:    cpi  $e7
0F40:    rnc

0F41:    mov  a,m
0F42:    cmp  c
0F43:    rnc

0F44:    cmp  b
0F45:    rc

0F46:       call $0cc4
0F49:    lxi  d,$0d04   #
0F4C:    dcx  h         # H--
0F4D:    dcx  h
0F4E:    jmp  $0ead



0F11, 0F80
-----------------------------------------------------collision detection?
#  D topleft screenpos spaceship (43e2:43e3)
#  BC = #rows#columns to check on enemy collision chars
(0f6e column-loop)
0F56:    push b
0F57:    push d
(0f65 row-loop)
0F58:     ldax d            # points to char composite of spaceship, fe:419b/419b
0F59:     cpi  $60          # char value of a smallbird is from 60 till C0
0F5B:     jc   $0f63        # go one row lower to check

0F5E:     cpi  $c0          # smallbird chars between 60 up until C0
0F60:     jc   $0cf4        # we have detected spaceship-smallbird object collision
(0f5b)
0F63:     inx  d
0F64:     dcr  b            # nr of rows to check (02)
0F65:     jnz  $0f58        # next row

0F68:    pop  d
0F69:    pop  b

0F6A:       call $0217      #   cursor met 1 kolom positie naar rechts
0F6D:    dcr  c            # nr of columns to check (02)
```

```
0F6E:    jnz  $0f56         # next column
0F71:    ret


0f06
-----------------------------------------------------collision detection, shield on?
0F74:    mvi  l,$e2         #   43e2
0F76:    mov  d,m
0F77:    inr  l
0F78:    mov  e,m           # DE (43e2:43e3) topleft cursorpos shapeship?
0F79:       call $0217      #   cursor met 1 kolom positie naar rechts
0F7C:    dcx  d
0F7D:    lxi  b,$0404       # #rows#columns to check (spaceship and shield area?)
0F80:       call $0f56      # collision detection?
0F83:    rz                 # no

0F84:    nop                # yes there was a collision
0F85:    nop
0F86:    ldax $43c2         #       v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
0F89:    sui  $0e
0F8B:    mov  b,a
0F8C:    adi  $2d
0F8E:    mov  c,a
0F8F:    lxi  h,$4b70       #       v4b70_wingformation-base
(0f9f)
0F92:    mov  a,m
0F93:    inr  l
0F94:    inr  l
0F95:    ani  $08
0F97:       cnz  $0fa6
0F9A:    inr  l
0F9B:    inr  l
0F9C:    mvi  a,$b0
0F9E:    cmp  l
0F9F:    jnz  $0f92         # until 4bb0
0FA2:       ret


0f97::
-----------------------------------------------------
0FA6:    inr  l
0FA7:    mov  a,m
0FA8:    dcr  l
0FA9:    cpi  $ca
0FAB:    rc

0FAC:    cpi  $ef
0FAE:    rnc

0FAF:    mov  a,m
0FB0:    cmp  c
0FB1:    rnc

0FB2:    cmp  b
0FB3:    rc

0FB4:    lxi  d,$0d02
0FB7:    dcx  h
0FB8:    dcx  h
0FB9:    jmp  $0ead



21ae, 21c2, 346a
------------------------------------------------------ dead-anim pointer assignment
# set pointer to a dead-anim slot
0FC0:    lxi  h,$4370       # Dead-anim slot #0/3
0FC3:       call $0fd8
0FC6:    lxi  h,$4374       # Dead-anim slot #1/3
0FC9:       call $0fd8
0FCC:    lxi  h,$4378       # Dead-anim slot #2/3
```

```
0FCF:       call $3758
0FD2:    lxi  h,$437c       # Dead-anim slot #3/3
0FD5:    jmp  $3758


0FC3, 0FC9
-------------------------------------------------
# dead anim (#0/#1 handling)
0FD8:    mov  a,m           #       4370/4374 afhankelijk van aanroeper (see 0fc0 code)
0FD9:    ana  a             # (Z when a=0, otherwise NZ)
0FDA:    rz                 #
                            #
0FDB:    mov  b,m           #
0FDC:    dcr  m             #       4370/4374 --1
0FDD:    inr  l             #
0FDE:    inr  l             #
0FDF:    mov  d,m           #
0FE0:    inr  l             #
0FE1:    mov  e,m           #
0FE2:    nop                #
0FE3:       call $0210      #       cursor met 1 kolom positie naar links
0FE6:    mov  a,b           #
0FE7:    ani  $0e           #       0000.xxx0
0FE9:    rrc                #       0000.0xxx
0FEA:    adi  $b0           #       1011.0xxx
0FEC:    mov  l,a           #
0FED:    mvi  h,$17         #       17b0+x
0FEF:    mov  l,m           #       17b0 -- 17b7
0FF0:    xchg               #       HL <-> DE
0FF1:    lxi  b,$ffdf       #           ( - $21 )
0FF4:    jmp  $3540         #       6 bytes copy DE->HL

    # ???
    # 0FF7: 68         mov  l,b
    # 0FF8: 3E 05      mvi  a,$05
    # 0FFA: 32 96 43   stax $4396          #       4396 := 05
    # 0FFD: C3 A4 0E   jmp  $0ea4
=============================================================================================
ENEMY (4b70-4baf) SPRITE MOVEMENT offsets (accessed by routine 0d70) pointers to are at 4b50-4b6f
#1000:0,20,64,a8,d4   1100:0,30,60,a4,d0    1200:0,44,88,ca   1300:0,28,54,9b,d0
    1000:  [01 01 01 01 02 02 02 02 02 02 02 02 01 01 01 01     0:list 1000 - 1010  #smallbird regular formation
    1010:  00] FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    (1010 <- 386B)
    1020:  [10 11 12 13 10 1D 0D 0E 0B 0C 0D 0E 0B 0C 06 06     1:list 1020 - 1060   (<- 0C1B)
    1030:  1E 03 1F 05 1C 04 1D 06 1E 03 03 03 03 03 1F 1C
    1040:  1D 1E 03 03 03 03 03 1F 05 1C 04 1D 06 1E 03 1F
    1050:  05 05 05 05 05 05 05 05 05 05 1C 04 04 11 12 13
    1060:  00] FF FF FF [0B 1E 19 06 06 06 06 06 06 1E 1F 1C    2:list 1064 - 10a4
    1070:  1D 06 06 06 06 06 1E 03 1F 05 1C 04 1D 06 06 1A
    1080:  04 1B 05 18 19 06 1A 04 1B 05 05 1C 04 1D 06 1E
    1090:  03 1F 05 05 05 05 05 1C 1D 1E 1F 05 05 05 05 05
    10A0:  05 05 18 1F 00] FF FF FF [10 04 04 1D 0D 0E 0B 0C    3:list 10a8 - 10d0
    10B0:  0D 0E 01 01 01 01 01 01 01 01 05 05 05 05 05 1C
    10C0:  04 04 1D 06 06 1E 03 03 1F 05 05 05 1C 11 12 13
    10D0:  00] FF FF FF [0B 0C 0D 0E 0B 0C 0D 0E 0B 0C 1A 1B    4:list 10d4 - 10fc
    10E0:  05 18 19 06 0D 0E 01 01 01 01 01 01 01 01 05 05
    10F0:  1C 1B 05 05 1C 04 1B 05 05 1C 04 1B 00] FF FF FF

    1100:  [0B 0C 0D 0E 0B 0C 09 09 09 09 0A 0A 09 09 0A 09     5:list 1100 - 112b
    1110:  16 17 14 07 07 07 1C 04 1D 06 1E 03 1F 05 1C 08
    1120:  08 08 08 08 08 08 08 05 05 05 05 00] FF FF FF FF
    1130:  [0B 0C 0D 0E 0B 0C 0A 0A 0A 0A 09 09 0A 0A 09 0A     6:list 1130 - 115b
    1140:  12 13 10 08 08 08 18 07 07 07 07 05 1C 04 1D 06
    1150:  1E 03 1F 07 07 07 07 05 05 05 05 00] FF FF FF FF
    1160:  [1C 04 04 04 1D 06 0D 0E 0B 0C 06 06 1E 15 16 17     7:list 1160 - 11a0
    1170:  14 19 06 1A 04 1D 06 1E 03 19 06 1A 04 1D 1E 03
    1180:  1F 1C 04 1B 05 18 03 1F 05 1C 04 1B 05 18 03 15
    1190:  16 17 14 1F 05 05 05 05 05 05 05 1C 04 1D 1A 1B
    11A0:  00] FF FF FF [0B 0C 0D 0E 0B 0C 0D 0E 0B 0C 0D 0E    8:list 11a4 - 11cc
    11B0:  02 02 02 02 02 02 02 02 05 05 18 03 19 1A 04 1B
    11C0:  05 18 03 1F 05 18 03 1F 05 05 18 1F 00] FF FF FF
```

```
    11D0:  [0B 0C 0D 0E 0B 0C 06 06 09 09 09 0A 09 09 0A 09     9:list 11d0 - 11fd
    11E0:  09 09 06 1A 04 11 12 13 10 08 08 08 07 07 07 08
    11F0:  08 08 05 05 05 05 05 05 05 05 05 05 05 00] FF FF

    1200:  [1C 11 12 13 10 04 1D 0D 0E 0B 0C 0D 0E 0B 0C 1E     a:list 1200 - 1240
    1210:  1F 05 18 19 0D 0E 0B 0C 1E 1F 05 05 05 05 05 18
    1220:  19 0D 0E 0B 0C 06 1E 1F 05 05 05 05 18 19 06 1E
    1230:  1F 05 05 05 05 05 05 05 05 1C 04 04 1D 1A 04 1B
    1240:  00] FF FF FF [18 03 03 19 06 06 06 06 06 06 06 06    b:list 1244 - 1284 #diving smallbird pattern bigboss level
    1250:  06 06 06 06 1A 04 1B 05 1C 04 1D 06 1E 03 03 19
    1260:  06 1A 04 04 04 1B 05 18 03 03 1F 05 1C 04 1D 06
    1270:  1A 04 1B 05 05 05 05 05 05 05 05 05 05 05 05 18
    1280:  03 19 1E 1F 00] FF FF FF [0B 0C 1A 1D 1E 03 19 06    c:list 1288 - 12c8
    1290:  1A 04 04 1D 06 1E 03 03 03 19 06 06 1A 04 04 04
    12A0:  04 1D 06 06 1E 03 03 03 03 03 03 1F 05 05 1C 04
    12B0:  04 04 04 1B 05 05 18 03 03 03 1F 05 1C 04 04 1B
    12C0:  05 18 03 1F 1C 1B 05 05 00] FF [18 03 19 06 06 06    d:list 12ca - 12ff
    12D0:  06 06 06 1A 1D 1E 19 1A 1D 06 1E 19 06 1E 15 16
    12E0:  17 14 07 07 07 08 08 08 08 05 05 18 03 03 19 06
    12F0:  06 1A 04 04 1B 08 08 08 08 05 05 05 05 18 1F 00]

    1300:  [0B 0C 0A 0A 09 09 09 0A 0A 09 09 09 0A 09 09 16     e:list 1300 - 1324
    1310:  17 14 07 07 07 08 08 08 08 07 07 08 08 08 08 07
    1320:  08 11 12 13 00] FF FF FF [0B 0C 09 09 0A 09 09 0A    f:list 1328 - 134e
    1330:  0A 0A 0A 09 0A 0A 0A 12 13 10 04 04 04 1B 18 03
    1340:  03 07 07 08 08 07 07 08 08 07 07 07 07 07 00] FF

    1350:  FF FF FF FF [1C 11 12 13 10 1D 0D 0E 0B 0C 09 0A     10:list 1354 - 1399 #diving smallbird pattern initial level
    1360:  09 09 0A 09 09 09 06 1A 04 1B 05 18 03 19 09 09
    1370:  0D 0E 0B 0C 0D 0E 02 02 02 02 02 02 02 02 02 02
    1380:  02 02 08 07 07 08 07 07 08 08 07 07 07 07 07 05
    1390:  05 05 05 05 1C 11 12 13 00] FF FF [0B 0C 0D 0E    11:list 139b - 13cd
    13A0:  0B 0C 0D 0E 0B 0C 1A 1D 06 1E 19 06 06 1A 04 1B
    13B0:  1C 04 1D 1A 04 1B 1C 04 1D 1A 04 1B 05 18 07 07
    13C0:  07 08 08 07 07 07 07 08 08 07 07 07 07 00] FF FF
    13D0:  [14 03 19 0D 0E 0B 0C 0A 0A 0A 09 0A 0A 0A 09 0A     12:list 13d0 - 13fb
    13E0:  0A 0A 06 1E 15 16 17 14 03 1F 05 05 08 07 07 07
    13F0:  08 07 07 07 08 08 05 05 05 05 05 00] FF FF FF FF

"  example:  small bird #5 16 bits pointer at [4b58,4b59] points to address 1354 gives $1c
       at code 0D30  base 1700 with offset 2* $1c = 1738 delivers the value $fc (two complement -4)
       which will be added to the column position of the smallbird [wxYz] -> [wx Y+-4 z],
       for row pos, 1738+1 = 0 [wxyZ] -> [wxy Z+0],
       because the position ranges from bit #7-3 for the 26 posible cols, only a value > +/- 8 shifts the object
     1354 =1c -> 1738 = -4, 1739 = 0
     1355 =11 -> 1722 = -4, 1723 = 0
     1356 =12 -> 1724 = -4, 1725 = 0
     ..
"
-------------------------------------------------------------------------------------------------------------------
SPRITE ANIM TABLE: SpaceShip and Small enemy Birds
spaceship and enemy OBJECTs (composition and animation sequence)
wingdata.element2 = object offset (see 1600 for offset values for ship anim)

#1600:  [10 14 18 1C 00 04 08 0C] ship-anim offset@1400

#SHIP SPRITES
#      0              04             08             0c                 SHIP anim seq  (8 objects)
1400:  [30 40 31 41]  [32 42 33 43]  [34 44 35 45]  [36 46 37 47]           #0/1/2/3
#       10             14             18             1c
1410:  [38 48 39 49]  [3A 4A 3B 4B]  [3C 4C 3D 4D]  [3E 4E 3F 4F]           #4/5/6/7

#1600:  10 14 18 1C 00 04 08 0C  [20 22 24 26 28 2A 2C 2E    ship-anim offset@1400
#1610:  30 32 34 36 38 3A 3C 3E   40 42 44 46 5C 5C 5E 5E]   ex: 1600=10 gives 1410 for [38 48 39 49]

"
@1400 animated FG objects

  SHIP ANIM:  8 objects
    object as visually positioned on screen (not the byte order written to videoram)
```

```
    #0        #1        #2        #3        #4        #5        #6        #7
    [30 31]   [32 33]   [34 35]   [36 37]   [38 39]   [3a 3b]   [3c 3d]   [3e 3f]
    [40 41]   [42 43]   [44 45]   [46 47]   [48 49]   [4a 4b]   [4c 4d]   [4e 4f]
===== -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  SMALL BIRDs: 39 objects @(1400+20)
    -- [L R] sprites --JT3:0788----------     -- single byte sprites ----------------------
    #0        #1        #2        #3          #4    #5    #6    #7    #8    #9   #a   #b
    [60 61]   [62 63]   [64 65]   [66 67]     [68]  [69]  [6a]  [6b]  [6c]  [6d] [6e]  [6f]

    -- [L R] sprites ------------------------------
    #c        #d        #e        #f        #10
    [70 71]   [72 73]   [74 75]   [76 77]   [7a 7b]

    -- 4 byte sprites --------      -----[T] [B]-- sprites JT3:07AA ------------------    --4 byte sprites--
    #11       #12       #13         #14  #15  #16   #17   #18   #19     #1a+1c  #1b+1d    #1e
    [80 81]   [82 83]   [84 85]     [86] [87] [88]  [89]  [8a]  [8b]     [8c]  [8d]       [8e 8f]
    [90 91]   [92 93]   [94 95]     [96] [97] [98]  [99]  [9a]  [9b]      [9c]  [9d]      [9e 9f]

    -- 4 byte sprites ------ ...
    #1f       #20       #21       #22       #23         #24         #25         #26
    [a0 a1]   [a2 a3]   [a4 a5]   [a6 a7]   [a8 a9]     [aa ab]     [ac ad]     [ae af]
    [b0 b1]   [b2 b3]   [b4 b5]   [b6 b7]   [b8 b9]     [ba bb]     [bc  ?]     [be bf]
===== -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
   Explosions 21 objects
    #0,1 #2,3 #4,5 #6,7  #8       #9        #a              #b                #c,d #e,f
    [c0] [c1] [c2] [c3]  [c4 c5]  [c6 c7]   [c8 c9 ca]      [cb cc cd]          [ce] [cf]
     [d0] [d1] [d2] [d3] [d4 d5]  [d6 d7]   [d8 d9 da]      [db dc dd]           [de] [df]

    #10  #11  #12  #13
    [e0] [e1] [e2] [e3]
=====
   Barriers for SHIP
                         [e4  e5   e6 e7]   [e8 e9 ea eb]       [ec ed ee ef]
    [f0 f1       f2 f3]                     [f8       fb]       [fc       ff]
                         [f4 f5    f6 f7]      [f9 fa]             [fd fe]

the objects itself also do undergo animations, the index value points to the byte sequence in mem
offset 3b below means a smallbird with following value in memory [8A 9A]  smallbird[Top, Bottom]
"

#SMALLBIRD SPRITES
#       20      22      24      26       28      2a      2c      2e      smallBIRD anim seq      pos
1420:  [60 61] [62 63] [64 65] [66 67]  [69 00] [69 00] [7A 7B] [7A 7B]      #0/1/2/3/5/5/16/16   [L][R]
#       30      32      34   35   36   37   38      39      3b      3e
1430:  [6B 00] [6B 00] [8C] [8D] [8C] [8D] [68 00] [68 00] [8A 9A] [8A 9A]   #7/7/26/27/26/27/4/4   [T]
#       40      42      44      46      48      4a      4c      4e                                    [B]
1440:  [6A 00] [6A 00] [8B 9B] [8B 9B] [68 00] [6B 00] [6A 00] [69 00]       #6/6/25/25/4/7/6/5    [T]
#       50      52      54      56      58      5a      5c      5e                                    [B]
1450:  [76 77] [74 75] [72 73] [70 71] [68 00] [86 96] [69 00] [87 97]    #15/14/13/12/4/20/5/21  [L][R]
#       60      62      64      66      68            6c
1460:  [6A 00] [88 98] [6B 00] [89 99] [68 00] 00 00 [A2 B2 A3 B3]             #6/22/7/23/4/32      [T]  diving
#       70              74              78            7c                                            [B]
1470:  [69 00] 00 00  [A4 B4 A5 B5]  [6A 00] 00 00 [A6 B6 A7 B7]               #5/33/6/34         [TL][TR]
#       80              84                          8c                                            [BL][BR]
1480:  [6B 00] 00 00  [A8 B8 A9 B9] FF FF FF FF [8A 9A] 00 00                  #7/35/24
1490:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
#       a0                          a8            ac
14A0:  [8B 9B] 00 00 FF FF FF FF  [8E 9E 8F 9F] [A0 B0 A1 B1]                 #25/30/31
14B0:  00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
#       c0              c4            c8            cc
14C0:  [9C 00] 00 00  [84 94 85 95] [82 92 83 93] [80 90 81 91]               #28/19/18/17
#       d0              d4            d8            dc
14D0:  [9D 00] 00 00  [AE BE AF BF]  [AC BC AD] 00 [AA BA AB BB]              #29/38/37/36


------------------------------------------------------------ initial values fleetformatons (and sounds?)
    14E0:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
    14F0:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

@1500   # preset values for the small birds (<- 05EC)
    1500:  08 6C 09 60  08 6C 09 60  08 6C 09 60  08 6C 09 60   preset wingdata, 4b70 from PC 0610 disasm
```

```
    1510:  08 6C 09 60  08 6C 09 60  08 6C 09 60  09 60 09 60

        # enemy slots movement table pointers (:= 1000)
    1520:  10 00  10 00  10 00  10 00  10 00  10 00  10 00  10 00   init sprite movement indexes (to $1000)
    1530:  10 00  10 00  10 00  10 00  10 00  10 00  10 00  10 00

------------------- Initial fleet positions, per gamelevel,  see table above: PC 0650 disasm
    1540:  50 20 70 20  60 28 60 38  50 40 70 40  40 38 80 38
    1550:  30 30 90 30  20 38 A0 38  18 48 A8 48  60 48 60 58

    1560:  60 48 60 58 48 58 78 58 38 50 88 50 28 48 98 48
    1570:  18 40 A8 40 18 30 A8 30 28 28 98 28 38 20 88 20

    1580:  60 20 50 20 70 20 40 28 80 28 30 30 90 30 20 38
    1590:  A0 38 60 58 50 58 70 58 40 58 80 58 30 58 90 58

    15A0:  60 20 50 28 70 28 40 30 80 30 30 38 90 38 20 40
    15B0:  A0 40 60 58 50 58 70 58 40 50 80 50 30 48 90 48

    15C0:  60 58 50 50 70 50 60 48 40 48 80 48 50 40 70 40
    15D0:  40 38 80 38 30 30 90 30 20 28 A0 28 10 20 B0 20

    15E0:  60 20 50 28 70 28 40 30 80 30 30 38 90 38 20 40
    15F0:  A0 40 60 20 50 28 70 28 40 30 80 30 30 38 90 38


-------- animation sequences
# offsets@1400   (1400 + 1c) -> #7 ship-obj
# for the spaceship @1600 this means we see the animation sequences 10 14 18 .. 0c
@1600    (<-08F0)                  @1608
    1600:  [10 14 18 1C 00 04 08 0C] [20 22 24 26 28 2A 2C 2E  first block is SPACE SHIP, second smallbirds
    1610:  30 32 34 36 38 3A 3C 3E 40 42 44 46 5C 5C 5E 5E    smallbirds

@1620   (<-0956)
    1620:  [50 51 52 53 54 55 56 57] FF FF FF FF FF FF FF FF   laser bullets 8x pixel shifted

@1630
    1630:  [48 48 50 50 4A 4A 52 52 4C 4C 54 54 4E 4E 56 56     boss-ship anim chars?
    1640:  48 48 56 56 4E 4E 54 54 4C 4C 52 52 4A 4A 50 50
    1650:  68 68 6C 6C 70 70 74 74 78 78 7C 7C 80 80 84 84
    1660:  68 68 84 84 80 80 7C 7C 78 78 74 74 70 70 6C 6C
    1670:  58 58 5A 5A 5C 5C 5E 5E 60 60 62 62 64 64 66 66
    1680:  78] FF [A0] FF FF [A8] FF [AC C0] FF [C8] FF FF [C4] FF [CC
    1690:  D0] FF [D8] FF FF [D4] FF [DC] FF FF FF FF FF FF FF FF


@16a3   (a b c) triplets  used at 0D9C
#                    0          1          2          3          4
    16A0:  FF FF FF  (01 02 08) (01 02 08) (01 02 0C) (01 02 10) (03     /
#                   5          6          7          8          9
    16B0:  04 14)  (03 04 18) (04 01 88) (04 01 90) (04 01 80) (04 01
#               a          b          c          d
    16C0:  80) (03 04 70) (03 04 74) (03 04 78) (03 04 7C)  FF FF FF


    16D0:  (01 02 30) (01 02 34) (01 02 38) (01 02 3C) (01 02 40) (01
    16E0:  02 44) (01 02 48) (01 02 4C) (04 04 50) (04 04 54) (04 04
    16F0:  58) (04 04 5C) (04 04 60) (04 04 64) (04 04 68) (04 04 6C)

@1700
    1700:  FF FF [01 00] FF [00 04 00 FC 00 00 FC 00 04 04 FE
    1710:  FC FE 04 02 FC 02 00 04 00 04 00 04 00 04] FF FF
    1720:  [FC 00 FC 00 FC 00 FC 00 04 00 04 00 04 00 04 00
    1730:  04 FC 04 04 FC 04 FC FC FC FC FC 04 04 04 04 FC             #13
    1740:  08 00 00] FF [01 00 F8] FF [08 01 02] FF [04 00 FA] FF     stepvalues (see 0e2a)
    1750:  [08 01 04] FF [08 00 FC] FF [08 05 06] FF [08 00 FE] FF
    1760:  [10 10 88 88 10 10 10 10] FF FF FF FF FF FF FF FF
    ------------------------------------------------------

    1770:  EC FC FD F4 ED 30 40 F5 EE 31 41 F6 EF FF FE F7      ship 1 + barrier shield band 1  (<- 0AB8)
```

```
1780:  E8 F8 F9 F0 E9 30 40 F1 EA 31 41 F2 EB FB FA F3      ship 1 + barrier shield band 2
1790:  E8 F8 F9 F0 E9 E4 E6 F1 EA E5 E7 F2 EB FB FA F3      ship 2+ barrier shield 2
17A0:  00 00 00 00 00 E4 E6 00 00 E5 E7 00 00 00 00 00      ship 2 within barrier surrounded

17B0:  F0 CA C4 BE B8 BE B8 BE C8 D8 C9 D9 CA DA CB DB      explosion of?
17C0:  CC DC CD DD C0 C1 C1 C2 00 C0 00 00 00 C3 00 00      small bird explosion
17D0:  C4 D4 C5 D5 C3 C3 C3 C3 C6 D6 C7 D7 FF FF FF FF      big bird explosion

17E0:  FF  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
17F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      (<-0AB0)
-------------------------------------------------------
```

TEXTSTRINGS format: <screenpos-2bytes> <FFFFFFFF -> markers-6bytes> <26chars-screenrowsize=1A>

```
1800:  (4320) FFFFFFFF 00 13 03 0F 12 05 21 00 00 08    "         SCORE1 H"   (<--0012)
1810:  09 2B 13 03 0F 12 05 00 00 13 03 0F 12 05 22 00    "I-SCORE  SCORE2 "
1820:  (4321) FFFFFFFF 00 20 20 20 20 20 20 00 00 00    "       000000   "
1830:  20 20 20 20 20 20 00 00 00 20 20 20 20 20 20 00    "000000   000000 "
1840:  (4322) FFFFFFFF 00 00 00 7F 20 00 00 00 00 00
1850:  03 0F 09 0E 20 20 00 00 00 00 00 7F 20 00 00 00    "COIN00          "
1860:  (4325) FFFFFFFF 00 00 00 00 00 00 00 09 0E 13     "              INS"
1870:  05 12 14 00 00 03 0F 09 0E 00 00 00 00 00 00 00    "ERT  COIN        "
1880:  (4327) FFFFFFFF 00 00 00 1F 00 21 10 0C 01 19     "          * 1PLAY"
1890:  05 12 00 00 00 21 03 0F 09 0E 00 00 1F 00 00 00    "ER   1COIN  *    "
18A0:  (4329) FFFFFFFF 00 00 00 1F 00 22 10 0C 01 19     "          * 2PLAY"
18B0:  05 12 13 00 00 22 03 0F 09 0E 13 00 1F 00 00 00    "ERS  2COINS *   "
18C0:  (432E) FFFFFFFF 00 00 00 13 03 0F 12 05 00 01    "   SCORE A"
18D0:  16 05 12 01 07 05 00 14 01 02 0C 05 00 00 00 00    "VERAGE TABLE"
18E0:  (4330) FFFFFFFF 00 00 00 00 00 00 00 00 22 20
18F0:  00 24 20 00 28 20 00 00 00 00 00 00 00 00 00 00
1900:  (4333) FFFFFFFF 00 00 00 00 00 00 00 00 22 20
1910:  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1920:  (4336) FFFFFFFF 00 00 00 00 00 00 00 00 25 20
1930:  00 21 20 20 00 2F 1B 21 20 20 2B 28 20 20 1C 00
1940:  (4339) FFFFFFFF 00 00 00 00 00 00 00 00 21 20       "10"
1950:  20 20 2B 29 20 20 20 00 00 00 00 00 00 00 00 00  "00-9000"
1960:  (433C) 00000000 00 00 00 00 00 00 00 00 00 00
1970:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1980:  (433D) 21002100 00 2C 00 21 29 28 20 00 14 01
1990:  09 14 0F 00 03 0F 12 10 0F 12 01 14 09 0F 0E 00
19A0:  (433E) FFFFFFFF 00 00 00 00 00 00 00 00 00 00
19B0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
19C0:  (4328) FFFFFFFF 00 00 00 00 00 00 00 00 00 00
19D0:  00 10 15 13 08 00 00 00 00 00 00 00 00 00 00 00
19E0:  (432C) FFFFFFFF 00 00 00 00 0F 0E 0C 19 00 21
19F0:  10 0C 01 19 05 12 00 02 15 14 14 0F 0E 00 00 00

1A00:  (4328) FFFFFFFF 00 00 00 00 00 00 00 00 07 01   "              GA"   (<--0B6A)  game over  ;-(      (<-0B6A)
1A10:  0D 05 00 00 0F 16 05 12 00 00 00 00 00 00 00 00   "ME  OVER        "

                                                      'PHOENIX' text depicted as smallbirds sprites
1A20:  (4328) 00FFFFFF 64 65 64 65 64 65 60 61 00 00     |********                **
1A30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 78 79   |
1A40:  (4329) FFFFFFFF 64 65 00 00 00 00 64 65 00 00     |**    **             **
1A50:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 7C 7D   |
1A60:  (432A) FFFFFFFF 64 65 64 65 64 65 60 61 00 00     |********
1A70:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   |
1A80:  (432B) FFFFFFFF 64 65 00 00 00 00 00 00 00 00     |**
1A90:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1AA0:  (432C) FFFFFFFF 64 65 00 68 00 68 00 68 68 68     |** * * *** *** ** * * * *  *
1AB0:  00 68 64 65 00 62 63 00 68 00 68 00 68 00 00 68   |
1AC0:  (432D) FFFFFFFF 64 65 00 68 00 68 00 68 00 68     |** * * * * *   ** * * ****
1AD0:  00 68 00 00 00 68 9D 00 68 00 68 00 76 77 70 71   |
1AE0:  (432E) FFFFFFFF 64 65 00 68 68 68 00 68 00 68     |** *** * * *** **** *  **
1AF0:  00 68 62 63 00 68 76 77 68 00 68 00 00 64 65 00   |
1B00:  (432F) 00000000 64 65 00 68 00 68 00 68 00 68     |** * * * * *   * ** * ****
1B10:  00 68 00 00 00 68 00 9D 68 00 68 00 74 75 72 73   |
1B20:  (4330) FFFFFFFF 64 65 00 68 00 68 00 68 68 68     |** * * *** *** * ** * *  *
1B30:  00 68 64 65 00 68 00 66 67 00 68 00 68 00 00 68   |

1B40:  [6C 6D 6E 6F] FF FF FF FF [6C 6D 6E 6F 64 65 66 67         (<--23A9) bigboss belt
```

```
    1B50:  63] FF [63 61 67] FF [67 65 6B] FF [6B 69 6F] FF [6F 6D    (1b50 <- 2034)

    1B60:  80 83 83 85 81 8C 8C 86 81 8C 8C 86 82 84 84 87   |
    1B70:  00 89 89 00 88 8D 8D 8B 88 8D 8D 8B 00 8A 8A 00   |
    1B80:  00 00 00 00 00 80 85 00 00 82 87 00 00 00 00 00   V

    1B90:  1B 80 1B 70 1B 60 1B 70 17 F0 17 F0 17 F0 17 F0  index [1b80] [1b70] [1b60] [1b70]  [17f0] [17f0] [17f0] [17f0]

    1BA0:  43 2C 00 00 00 00 00 00 00 21 00 0F 12 00 22 10  [432c]
    1BB0:  0C 01 19 05 12 13 00 02 15 14 14 0F 0E 00 00 00

    1BC0:  41 54 76 7E  - 42 55 77 7F - 41 56 74 7C - 42 57 75 7D       BOSS spaceship (interior) objdata?
    1BD0:  44 51 72 7A  - 45 52 73 7B - 46 51 70 78 - 47 52 71 79
    1BE0:  41 51 70 78  - 42 52 71 79 - 41 51 72 7A - 42 52 73 7B
    1BF0:  41 51 74 7C  - 42 52 75 7D - 41 51 76 7E - 42 52 77 7F

    ------------------------------------------------------
SCREEN Boss-Level (229b, 2470)

    1C00:  00 01 00 06 00 02 03 04 00 01 00 08 00 02 03 04  bottom of starfield?
    1C10:  00 00 07 00 01 02 00 09 00 03 04 00 00 03 04 00
    1C20:  00 01 00 02 00 03 0A 00 04 00 00 01 02 00 06 00
    1C30:  03 04 00 00 01 00 02 00 03 00 04 00 03 05 00 00
    1C40:  00 00 07 00 01 00 02 00 00 05 00 00 03 00 04 01
    1C50:  02 00 03 00 08 04 00 01 02 06 00 03 00 04 00 02
    1C60:  01 02 03 00 05 00 00 04 00 01 02 00 00 03 04 0B
    1C70:  00 01 00 02 00 03 00 00 04 00 00 09 00 00 02 00
    1C80:  07 00 00 01 00 00 02 00 00 03 00 08 04 00 01 00
    1C90:  00 06 00 01 00 02 00 01 03 04 01 03 01 02 03 04
    1CA0:  00 05 00 01 02 00 09 00 03 04 00 01 00 01 02 03
    1CB0:  04 00 02 00 00 01 02 00 03 04 00 06 00 00 01 00
    1CC0:  00 01 02 00 05 00 00 03 00 04 00 07 00 01 00 02
    1CD0:  00 00 03 00 04 00 04 00 0A 00 01 00 02 00 03 00
    1CE0:  01 00 07 00 02 00 03 04 00 05 00 01 00 02 00 00
    1CF0:  08 03 04 00 01 00 02 00 03 00 04 00 00 06 00 03  top of starfield?

                                                              (1d09)     BOTTOM
    1D00:  0C 0D 0C 0F 07 07 01 00 00  [4C.4D.4E.4F.4F.4E.4D         boss-ship  line -9 (bottom)
    1D10:  4C]  00 00 1F 0E 06 0D 01 0E 05 08 0C 0E 0C 0A 00
                                                                               line -9 body right
    1D20:  00  [4D.4F.5E.5E.5E.5E.5E.5E.5E.5E.4F.4D]  00 00 06                 line -8
    1D30:  0B 0D 08 0E 03 02 00 01 00  [4C.4F.5E.5E.5E.5E.5E                   line -7  left
    1D40:  5E.5E.5E.5E.5E.5E.5E.4F.4C]  00 09 07 0A 03 04 00                   line -7  right
    1D50:  0A 00  [4D.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E                   line -6 body
    1D60:  5E.5E.5E.4D]  00 00 0E 0F 08 08 00  [5C.60.6A.60.6A                 line -5 band left
    1D70:  60.6A.60.6A.60.6A.60.6A.60.6A.60.6A.60.6A.5D]   00                       -5 band right
    1D80:  01 02 02 06 01 00 00 00  [58.59.5A.5B.5B.5B.7E.7F                   line -4 left
    1D90:  5B.5B.5B.4A.49.48]   00 00 00 03 0E 0B 0D 05 04 05                       -4 right
    1DA0:  0A 08 00 00  [58.59.5A.4B.76.77.4B.4A.49.48]  00 00                 line -3
    1DB0:  01 03 0F 02 03 00 00 03 03 07 02 0A 03 07 00 00
    1DC0:  [58.50.51.52.53.48]  00 00 0B 01 02 03 0F 0E 0C 02                 line -2
    1DD0:  05 0C 06 00 04 06 07 0E 0F 09 00  [40.41.42.43]  00       boss-ship  line -1 (top)
    1DE0:  07 03 0A 08 0D 00 09 0B 0C 0A                                          TOP

"BOSS-SHIP BITMAP example
                      [40.41.42.43]   <- top
                   [58.50.51.52.53.48]
              [58.59.5A.4B.76.77.4B.4A.49.48]
         [58.59.5A.5B.5B.5B.7E.7F.5B.5B.5B.4A.49.48]
[5C.60.6A.60.6A.60.6A.60.6A.60.6A.60.6A.60.6A.60.6A.5D]    <-- band
   [4D.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.4D]
      [4C.4F.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.5E.4F.4C]
            [4D.4F.5E.5E.5E.5E.5E.5E.5E.5E.4F.4D]
                  [4C.4D.4E.4F.4F.4E.4D.4C]    <-bottom>
"
    ------------------------------------------------------

                                      FF FF FF FF FF FF
    1DF0:  3A 1D 42 D6 01 C8 32 8F 43 FF FF FF FF FF FF FF

        1E00:  20 30 21 31 22 32 23 33 24 34 25 35 26 36 27 37
```

```
1E10:  28 38 29 39 2A 3A 2B 3B 2C 3C 2D 3D 2E 3E 2F 3F
" ????
spriral                 circular
galaxy  moon    astroid  galaxy
[20 21] [22 23] [24 25]  [26 27]   []
[30 31] [32 33] [34 35]  [36 37]
"

1E20:  49 48 4A 4B 4A 49 4A 49 48 4A 48 49 4B 48 4A 48  #pointers to FG videoram
1E30:   4A 49 4B 49 4B 4A 49 48 49 49 4A 4A 48 49 4A 48  # 494a, 4849, ...

1E40:  A0 60 40 00 E0 C0 C0 60 80 20 60 40 20 40 00 80   #  n * 20  offset values
1E50:  40 00 20 E0 00 60 00 A0 E0 20 80 00 C0 80 A0 E0

1E60:  [00 04 08 0C 10 14 18 1C] [00 08 10 18 04 0C 14 1C] # also offsets
1E70:  [00 0C 18 04 04 1C 08 14] [00 10 04 14 08 18 0C 1C]

1E80:  10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  #(1e80 <- 2052) used as offsets
1E90:  10 12 14 16 18 1A 1C 1E 11 13 15 17 19 1B 1D 1F

1EA0:  4A 4B 49 4A 48 4A 48 49 49 4A 49 4B 48 4B 4A 4A   #pointers to BG videoram
1EB0:   48 49 48 4A 48 48 49 4A 49 49 4A 48 4A 49 4B 48  # see 06B0

1EC0:  00 20 60 40 E0 80 20 60 40 A0 00 00 40 20 C0 20  # n * 20 offset values
1ED0:  A0 80 E0 40 60 C0 20 A0 E0 40 60 C0 20 40 20 80



21FC
================================================checksum [B] van copyright regel?
1EE0:    lxi  d,$433d       #   positie van de regel onder de 1980 Taito copyright
1EE3:    lxi  b,$001a       #   26 kolommen
(1eed)                      #
1EE6:    ldax d             #
1EE7:    add  b             #
1EE8:    mov  b,a           #
1EE9:       call $0217      #   cursor met 1 kolom positie naar rechts
1EEC:    dcr  c             #
1EED:    jnz  $1ee6         #

1EF0:    ldax d             #   3ffd -> b8
1EF1:    add  b             #   [A=47] := B8 + 8F
1EF2:    adi  $b9           #   [A=00] := 47 + B9
1EF4:    lxi  h,$4389       #   v4389_checksum?
1EF7:    add  m             #
1EF8:    mov  m,a           #
1EF9:    ret                #   0049: jmp 001a

    ------------------------------------------------------


starfield?
    1F00:  00 00 00 01 00 00 00 02 00 00 00 00 03 00 00 00
    1F10:  00 04 00 00 00 00 01 00 00 00 05 00 02 00 03 00
    1F20:  00 00 04 00 07 00 00 00 06 00 01 00 02 0C 00 03
    1F30:  04 00 00 01 00 08 00 00 02 00 0C 03 04 0E 00 00
    1F40:  00 01 02 00 0D 03 04 0F 01 0C 07 0A 02 0D 03 08
    1F50:  06 0C 04 09 05 0F 01 02 0D 03 0C 04 0D 05 0F 0C
    1F60:  01 02 0E 0C 03 0F 0D 05 0E 0D 0C 0F 0D 04 0C 01
    1F70:  0E 05 0F 0D 07 0C 06 0E 0D 0F 09 0C 0F 0D 0E 0D
    1F80:  02 0D 0C 0F 05 0E 0D 0C 0F 06 0E 0F 0C 0D 0F 0C
    1F90:  06 0D 04 0B 0C 0F 05 0D 05 03 0E 07 0C 0D 04 05
    1FA0:  01 02 0E 03 0C 04 0F 05 08 0C 07 01 0D 04 0E 02
    1FB0:  0C 01 0F 03 05 0D 00 0E 00 09 0C 06 0D 00 01 02
    1FC0:  01 02 03 00 00 0D 00 0A 00 00 00 0E 00 05 00 08
    1FD0:  00 0C 00 00 03 00 00 07 00 00 00 04 00 00 06 00
    1FE0:  00 00 00 01 00 00 00 00 02 00 00 00 00 03 00 00
    1FF0:  00 04 00 05 00 00 00 00 00 01 00 00 00 00 02 00


JT4:{02,06,16}<-080F  smallbird fight
============================================================ ================================
```

```
2000:       call $0876      # appeareance of spaceship
2003:       call $0df0      # laser projectile
2006:       call $24a0
2009:    lxi  h,$435f       #  v435f_8bitscountup
200C:    mov  a,m
200D:    ani  $03           #
200F:    mov  b,a           #       B = bits #0/1 (of countup)
2010:    inr  m             #       increase countup
2011:    ldax $43ba         #   v43ba_smallbird_fleetsize
2014:    ana  a
2015:    jz   $21ba

2018:    cpi  $05           #  still 5 birds left?
201A:    jnc  $2130         #  jump if so

201D:    dcr  l             #  435e
201E:    mov  a,b           #       B = bits #0/1 (of countup)
201F:    ana  a
2020:    jnz  $2025
2023:    mvi  m,$ff         #    435e := ff, if bits#0/1 ==00 (from v435f_8bitscountup)
(2020)                      # B > 0
2025:    mov  a,m           #  435e
2026:    ana  a
2027:    jz   $2130         #      == 0
202A:    jmp  $2146         #      otherwise jump



(23a2)
------------------------------------------------------
2030:    ani  $03
2032:    cpi  $01
2034:    lxi  d,$1b50       # rom ptr
2037:    jmp  $23ac



(06f3)
------------------------------------------------------plot on timed trigger?
2040:    lxi  h,$43af       #       v43af_scrolltrigger
2043:    ldax $43b9         #       v43b9_videoscroll
2046:    mov  c,a           #
2047:    cmp  m             #       vb  v43af_scrolltrigger = v43b9_videoscroll  ??
2048:    rnz                #           <> (skip if not)

#  v43af_scrolltrigger == v43b9_videoscroll
2049:    mov  a,m           #           43af_scrolltrigger (ex 40)
204A:    inr  l             # 43af->43b0      (3F)     v43b0_16bit_romptr = 3F00
204B:    sub  m
204C:    dcr  l             # 43af
204D:    mov  m,a           #           v43af_scrolltrigger =- 43b0
204E:    inr  l
204F:    inr  l
2050:    inr  m             # 43b1++  (3Fxx)
2051:    mov  a,m           #  [A] used as offset for 1e80 rombase
2052:    lxi  h,$1e80       # romdata entry (see above data chunk)
2055:    ani  $1f           #  offset capped to 1e9f
2057:    add  l
2058:    mov  l,a
2059:    mov  b,m           # [B :=  *(1e8x or 1e9x)]
205A:    adi  $20
205C:    mov  l,a           # HL + 20
205D:    mov  d,m           #   1EAx of 1EBx
205E:    adi  $20
2060:    mov  l,a           # HL + 20
2061:    mov  e,m
2062:    mov  a,c
2063:    rrc
2064:    rrc
2065:    rrc
2066:    ani  $1f
```

```
2068:    add  e
2069:    inr  a
206A:    mov  e,a
206B:    mov  a,b
206C:    stax d
206D:    ret

hexdata = "
    1E80:  10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
    1E90:  10 12 14 16 18 1A 1C 1E 11 13 15 17 19 1B 1D 1F

    1EA0:  4A  4B  49  4A  48  4A  48  49  49  4A  49  4B  48  4B  4A  4A      pointers to BG videoram
    1EB0:   48  49  48  4A  48  48  49  4A  49  49  4A  48  4A  49  4B  48     4A48, 4b49, 4948, ...

    1EC0:  00 20 60 40 E0 80 20 60 40 A0 00 00 40 20 C0 20
    1ED0:  A0 80 E0 40 60 C0 20 A0 E0 40 60 C0 20 40 20 80
    "

(0bc1)
------------------------------------------------------
# C=E-0a+c0, B=D+00
2070:    mov  a,e
2071:    sui  $0a
2073:    adi  $c0
2075:    mov  c,a
2076:    mov  a,d
2077:    aci  $00
2079:    mov  b,a
207A:    mov  a,m
207B:    lxi  d,$2800           # explosion pattern (begin)
207E:    lxi  h,$2900           # explosion pattern (end)
2081:    jmp  $2085

(2081,2426)
------------------------------------------------------ ship explosion preparation
2085:    sui  $20
2087:    rlc
2088:    rlc
2089:    nop
208A:    ani  $e0
208C:    mov  l,a
208D:    mvi  a,$e0
208F:    sub  l
2090:    mov  l,a

(20a8)                 # jump start
------------------------
2091:    mvi  a,$3f
2093:    sub  c
2094:    mvi  a,$43
2096:    sbb  b
2097:    jnc  $20b0     # ship explosion animation
# DE+=10, BC-=20
209A:    inx  h
209B:    inx  h
209C:    mov  a,e
209D:    adi  $10
209F:    mov  e,a
20A0:    mov  a,c
20A1:    sui  $20
20A3:    mov  c,a
20A4:    mov  a,b
20A5:    sbi  $00
20A7:    mov  b,a
20A8:    jmp  $2091     # jump back


(2097)
------------------------------------------------------ship explosion animation
# memaccess 2900-29ff range from 20b2
20B0:    push b
```

```
(20c9,20de)                #--- start and odd HL pointer processing
20B1:     mov  a,m
20B2:     xthl             #       HL <-> SP

#loop-b
20B3:      mvi  b,$08      # loop-init
(20c2)
20B5:     mvi  m,$00       #    clear mem
20B7:     rrc              # A bit#0 set?
20B8:     jnc  $20bf       # skip if not
#  explosion direct plot  sequence
20BB:     xchg             # DE<->HL
20BC:     mov  c,m
20BD:     xchg
20BE:     mov  m,c
(20b8)                     # -- even charbyte handling --
20BF:     inx  h
20C0:     inx  d
20C1:     dcr  b           #loop-b
20C2:     jnz  $20b5

20C5:     xthl             #       HL <-> SP  (fe 2913)
20C6:     inx  h
20C7:     mov  a,l         # odd pointer pos (fe HL=290F)
20C8:     rrc              # make carry true
20C9:     jc   $20b1       # and therefore a jump

                           #--- even HL pointer processing
20CC:     mov  a,l
20CD:     ani  $1f
20CF:     jz   $20e1
20D2:     xthl             #       HL <-> SP
20D3:     mov  a,l
20D4:     sui  $30
20D6:     mov  l,a
20D7:     mov  a,h
20D8:     sbi  $00
20DA:     mov  h,a
20DB:     xthl             #       HL <-> SP
20DC:     cpi  $3f         #   HL (h-part) becomes lower than 4000 (start of fgscreen)
20DE:     jnz  $20b1
(20cf)
20E1:    pop  b
20E2:    ret



0b12,0bc4,2412
------------------------------------------------------
20E8:    mov  b,a
20E9:    mov  a,d
20EA:    adi  $08
20EC:    mov  d,a          #   d+=8
20ED:       call $211c     # perform video scroll
20F0:    rrc
20F1:    rrc
20F2:    rrc
20F3:    add  e
20F4:    ani  $1f
20F6:    mov  c,a
20F7:    mov  a,e
20F8:    ani  $e0
20FA:    ora  c
20FB:    mov  e,a
20FC:    mov  a,b
20FD:    rrc
20FE:    rrc
20FF:    ani  $0e
2101:    adi  $90
2103:    mov  l,a
```

```
2104:    mvi  h,$1b         # 1b90+x romaddress
2106:    mov  a,m
2107:    inr  l
2108:    mov  l,m
2109:    mov  h,a
210A:    lxi  b,$0404
210D:    jmp  $0ad6         #  transfer block[B,C] of romdata(HL) to screen(DE)



20ED
------------------------------------------------------perform video scroll
211C:    lxi  h,$43b9       # v43b9_videoscroll
211F:    mov  a,m
2120:    cpi  $10
2122:    rc                 #   [0-9] return

2123:    cpi  $30
2125:    rnc                #   [>=30] return

2126:    mvi  a,$10
2128:    mov  m,a           # reset v43b9_videoscroll := 10
2129:    stax $5800         VIDEO Scroll Reg
212C:    ret



201a,2027
--------------------------------------------------
# bits #0/1 (of v435f_8bitscountup)
2130:    mov  a,b           # valued from bits#10 of the counter at 435F
2131:    ana  a
2132:    jz   $2150     # bits #10 == 0
2135:    cpi  $01       # bit #0 set
2137:    jz   $2160
213A:    cpi  $02       # bit #1 set
213C:    jz   $2170
213F:    jmp  $2180

    # ??
    # 2142: 90         sub  b
    # 2143: A5         ana  l
    # 2144: 50         mov  d,b
    # 2145: 60         mov  h,b


(202a)
--------------------------------------------------
2146:    mov  a,b
2147:    rrc
2148:    jnc  $2190
214B:    jmp  $21a5


(2132)
--------------------------------------------------
2150:       call $0a50      # plot enemy (smallbird) fleet?
2153:       call $3000      # Jumptable 6
2156:    jmp  $0f00         # barrier enemy collision?



(2137)
--------------------------------------------------
2160:       call $24c4
2163:       call $0c40
2166:       call $0d1c
2169:    jmp  $0fc0


(213c)
```

```
------------------------------------------------------
2170:       call $0d70
2173:    jmp  $2560


(213f)
------------------------------------------------------
2180:       call $24c4
2183:       call $0c40
2186:       call $0a6c          # Modify wingformation?
2189:    jmp  $0fc0


(2148)
------------------------------------------------------
2190:       call $0a50          # plot enemy (smallbird) fleet?
2193:       call $3000          # Jumptable 6
2196:       call $0f00          # barrier enemy collision?
2199:       call $2560
219C:    jmp  $0c40



(214b)
------------------------------------------------------
21A5:       call $0d1c
21A8:       call $0d70
21AB:       call $0a6c          # Modify wingformation?
21AE:       call $0fc0
21B1:    jmp  $24c4


(2015)
------------------------------------------------------ no more enemies in stage
21BA:    mov  a,b
21BB:    rrc
21BC:    jnc  $2204
21BF:       call $0c40
21C2:       call $0fc0
21C5:       call $24c4
21C8:    ldax $43b8         #    v43b8_staging0-F_videobit1
21CB:    ani  $0f           # maintain low-end nibble
21CD:    cpi  $0b           #  boss-ship smallbird fighting stage
21CF:    jc   $2204         #  return if not in fight mode yet
#---                        prepare for fighting sequence
21D2:    mvi  a,$10         #   smallbirds count (10)
21D4:    stax $43ba         #   v43ba_smallbird_fleetsize
21D7:    jmp  $0526



012b
------------------------------------------------------bigbird vogel hatch animatie in intro scherm
#  wingdata opzetten (zie ook info 34c0)
21DC:    mov  a,m           #   (v43989_16bits_counter) lowbyte
21DD:    nop                #
21DE:    mov  b,a           #
21DF:    lxi  h,$4b73       #   wingdata 1e enemy 4e element
21E2:    ani  $07           #
21E4:    mov  m,a           #   bepaal nieuwe waarde voor 4373 obv counter lowpart and 07
21E5:    dcr  l             #
21E6:    mvi  m,$ef         #   3e element (4b72) := EF
21E8:    dcr  l             #
21E9:    mvi  m,$49         #   2e element (4b71) := 49
21EB:    dcr  l             #
21EC:    mov  a,b           #
21ED:    ani  $f8           #       v43989_16bits_counter and xxxx.x000
21EF:    rrc                #       0xxx.xx00
21F0:    rrc                #        00xx.xxx0
21F1:    rrc                #        000x.xxxx
21F2:    adi  $3a           #
```

```
21F4:    mov  e,a          #
21F5:    mvi  d,$23        #       233a + 000x.xxxx
21F7:    ldax d            #
21F8:    mov  m,a          #   1e element (4b70) := xxxx
21F9:       call $34c0
21FC:    jmp  $1ee0



21bc,21cf,346d
--------------------------------------------------
2204:    lxi  h,$43b6
2207:    dcr  m
2208:    mov  a,m
2209:    cpi  $a0
220B:    rnc

220C:    mvi  l,$a4        #   v43a4_JT1-indexer
220E:    mvi  m,$02        #                   := 02
2210:    mvi  l,$a6        #   v43a6_barrier_repeat_timer
2212:    mvi  m,$00        #                   := 00
2214:    mvi  l,$b8        #   43b8
2216:    inr  m            #    :=  ++1
2217:    mov  a,m          #
2218:    ani  $0e          #
221A:    rrc               #
221B:    adi  $60          #
221D:    mov  e,a          #
221E:    mvi  d,$17        #   17xx
2220:    inr  l
2221:    inr  l
2222:    ldax d
2223:    ana  a
2224:    jp   $222a
2227:    inr  l
2228:    ani  $7f
222A:    mov  m,a
222B:    jmp  $0380        #   schoon foreground behalve HUD (1e 3 regels)



JT4:{08,0c,10}<-080F  square '*' window anim
--------------------------------------------------
2230:    lxi  h,$439c    #v439c increment counter?
2233:    mov  a,m
2234:    inr  m
2235:    nop
2236:    rrc
2237:    ani  $3f
2239:    cpi  $0d
223B:    jz   $2292

223E:    mvi  b,$1f
2240:    jc   $2260

2243:    mvi  b,$00
2245:    sui  $0e
2247:    cpi  $0d
2249:    jnz  $2260


(Jt4:{18,1a,1c,1e}<-080f)  something in smallbird + bigboss fight next stage?
--------------------------------------------------
224C:    lxi  h,$43b8      #       v43b8_staging0-F_videobit1
224F:    inr  m            # next stage
2250:    mvi  l,$a4        #       v43a4_JT1-indexer
2252:    mvi  m,$02        #                     := 02  (init vars?)
2254:    ret

?
--------------------------------------------------
```

```
    2255:    mov  e,b
    2256:    mvi  l,$a4         #       v43a4_JT1-indexer
    2258:    mvi  m,$02         #                       := 02
    225A:    ret


(2240,2249)
-----------------------------------------------------game-over asterix starfield
# called multiple times to draw an asterix (*) border from inside out
2260:    mov  c,a
2261:    rrc
2262:    rrc
2263:    rrc
2264:    mov  d,a
2265:    ani  $1f
2267:    mov  e,a
2268:    mov  a,d
2269:    ani  $e0
226B:    adi  $b0
226D:    mov  l,a
226E:    mov  a,e
226F:    aci  $41       #HL 41b0 screenA center location?
2271:    mov  h,a
2272:    mov  a,l
2273:    sub  c
2274:    mov  l,a
2275:    mov  a,c
2276:    inr  a
2277:    mov  c,a
2278:    rlc
2279:    mov  e,a
227A:    mov  d,c
(2280,228e)
227B:    mov  m,b
227C:    inx  h
227D:    mov  m,b
227E:    inx  h
227F:    dcr  d         #D-innerloop
2280:    jnz  $227b

2283:    mov  a,l
2284:    sub  c
2285:    sub  c
2286:    sui  $20
2288:    mov  l,a
2289:    mov  a,h
228A:    sbi  $00
228C:    mov  h,a
228D:    dcr  e         #E-outerloop
228E:    jnz  $227a
2291:    ret


(223b)
-------------------------------------------------------
2292:    lxi  h,$43b8       #  v43b8_staging0-F_videobit1
2295:    mov  a,m           #
2296:    ani  $08           #  game over staging to bigg-boss anims
2298:    jz   $22f0         #

229B:    lxi  h,$1c00       # SCREEN Boss-level
229E:    lxi  d,$4b3f       # VIDEO RAM Charset B (links-onder)
22A1:    mvi  b,$47
(22ad)
22A3:    mov  a,m
22A4:    stax d
22A5:    inr  l
22A6:    dcx  d
22A7:    mov  a,m
22A8:    stax d
22A9:    inr  l
```

```
22AA:    dcx  d
22AB:    mov  a,b
22AC:    cmp  d
22AD:    jnz  $22a3
22B0:    jmp  $22e0


JT4:{12}<-080F  anim appearing big-boss baseship rowscroll?
------------------------------------------------------ scrolling BossShip?
22B4:       call $067a          #   scroll background and plot starfield
22B7:    lxi  h,$43b4           #
22BA:    dcr  m                 #
22BB:    mov  a,m               #
22BC:    cpi  $28               #
22BE:    jnz  $0848             #
22C1:    mvi  l,$67             #   4367
22C3:    mvi  m,$ff             #       := FF
22C5:    ret



JT4:{14}<-080f  anim baseship row scrolldown?
------------------------------------------------------ scrolling BossShip?
22CA:    lxi  h,$43b4
22CD:    mov  a,m
22CE:    cpi  $c0
22D0:    jnz  $0834
22D3:    mvi  m,$30
22D5:    mvi  l,$67             #   4367
22D7:    mvi  m,$ff             #       := FF
22D9:    mvi  l,$bc             #   v43bc_?
22DB:    mvi  m,$3f             #       := 3F
22DD:    ret


(22b0)
------------------------------------------------------
22E0:    mvi  a,$71         #   harde waarde voor scroll reg
22E2:    stax $43b9         #   v43b9_videoscroll
22E5:    stax $5800         VIDEO Scroll Reg
22E8:    ret


(2298)
------------------------------------------------------
22F0:       call $03a0      #   schoon background met 00
22F3:    xra  a             #   eor a   [A:=0]
22F4:    jmp  $22e2


(24d8)
------------------------------------------------------boss-ship belt animation
# background boss-fight
22FA:    lxi  h,$4aaa           #left pos of belt boss-ship    (Video RAM Charset B range)
22FD:    mvi  b,$12         #b-loop counter
22FF:    ldax $488a             #right pos of belt boss ship
2302:    mov  c,a
(231c)
2303:    mov  a,c
2304:    ani  $03
2306:    rlc
2307:    rlc
2308:    mov  d,a
2309:    mov  c,m
230A:    mov  a,c
230B:    ani  $0c
230D:    rrc
230E:    rrc
230F:    ora  d
2310:    ori  $60
2312:    mov  m,a
```

```
2313:    mov  a,l
2314:    sui  $20          # 1 column to the right?
2316:    mov  l,a
2317:    jnc  $231b
231A:    dcr  h
(2317)
231B:    dcr  b
231C:    jnz  $2303
231F:    ret


(24db)
------------------------------------------------------ anim parts of boss-spaceship?
2322:    lxi  h,$43a7      # v43a7_romscreendata-index8
2325:    inr  m            #       increase index
2326:    mov  a,m
2327:    ani  $07          #       0-7 base-index
2329:    rlc               #
232A:    rlc               #
232B:    rlc               #       index for chunck of 8 objectbytes
232C:    adi  $c0
232E:    mov  l,a
232F:    mvi  h,$1b        #   HL 1bc0 + v43a7_romscreendata-index8 * 8
2331:    lxi  d,$49a6      #   background screen pos xy(12,6)
2334:    lxi  b,$0402      #   2x4 blok
2337:    jmp  $0ad6        # transfer block[B,C] of romdata(HL) to screen(DE)

# BOSS spaceship objdata to be copied to background screen
"
    1BC0:  [41 54 76 7E  42 55 77 7F]   [41 56 74 7C  42 57 75 7D]
    1BD0:  [44 51 72 7A  45 52 73 7B]   [46 51 70 78  47 52 71 79]
    1BE0:  [41 51 70 78  42 52 71 79]   [41 51 72 7A  42 52 73 7B]
    1BF0:  [41 51 74 7C  42 52 75 7D]   [41 51 76 7E  42 52 77 7F]
"
------------------------------------------------------
# HEXDATA offset vanuit 21F7 (ldax) voor opslag in 4b70 enemy#0 wingdata (misused for this)
# Bigbird hatch anim in intro screen
"
                                    0  1  2  3  4  5
    233A:                           01 02 03 04 05 06

            7  8  9  a  b  c  d  e  f 10 11 12 13 14 15 16
    2340:  07 0A 07 0A 07 0A 07 0A 07 0A 09 08 04 03 02 01

    2350:  FF
"

24ae, 24bc
------------------------------------------------------ (prelude) boss-ship being hit?
# boss-ship scene when spaceship is firing/about to fire?
2351:   ldax d          #43c8 rapid fire mode
2352:    ani  $08       #bit #3 set?
2354:    rz

2355:    mov  a,m       #?  43e6/e7  = SpaceShip laser projectile (top) ahead
2356:    inr  l
2357:    mov  l,m
2358:    adi  $08
235A:    mov  h,a
235B:    ldax $43b9     #       v43b9_videoscroll
235E:    rrc
235F:    rrc
2360:    rrc
2361:    add  l
2362:    ani  $1f
2364:    mov  b,a
2365:    mov  a,l
2366:    ani  $e0
2368:    ora  b
2369:    mov  l,a
236A:    mov  a,m       #actual position screenB left above space ship
```

```
236B:    mov  b,a
236C:    ani  $fc
236E:    cpi  $4c          #4c is a sprite for hull part of the bigboss ship
2370:    jz   $237b

2373:    ani  $f0          #  only high nibble
2375:    cpi  $60          #belt of the boss ship
2377:    jz   $2398
237A:    ret

(2370)
------------------------------------------------------ modify boss-ship after being hit
237B:    ldax d            # 43c4 (laser object spaceship position)
237C:    ani  $f7
237E:    stax d
237F:    mvi  a,$ff
2381:    stax $4366        #  v4366_BossShip-hit := FF (detected)
2384:    mov  a,b
2385:    dcr  a
2386:    mov  m,a
2387:    cpi  $4b              # boss-ship alien eyes
2389:    rnz

238A:    mvi  m,$00
238C:    dcr  l
238D:    mov  a,m
238E:    cpi  $5e              # boss-ship hull (untouched)
2390:    rnz

2391:    mvi  m,$4f
2393:    ret


(2377)
------------------------------------------------------
2398:    ldax d
2399:    ani  $f7
239B:    stax d            # @[43c4] (laser object spaceship position)  ####.0### (clear out bit #3)
239C:    inr  e
239D:    inr  e
239E:    ldax d            # 43c6 (specobj1#val3)
239F:    ani  $04
23A1:    mov  a,b
23A2:    jnz  $2030
23A5:    ani  $0c
23A7:    cpi  $04
23A9:    lxi  d,$1b40      #    music related? romaddress
(2037)                     # d=1b50
23AC:    jz   $23c0

23AF:    mov  a,b
23B0:    ani  $0f          # maintain low-end nibble
23B2:    add  e
23B3:    mov  e,a
23B4:    ldax d
23B5:    mov  m,a
23B6:    mvi  a,$ff
23B8:    stax $4366        # v4366_BossShip-hit
23BB:    ret


(23ac)
------------------------------------------------------
23C0:    dcr  l
23C1:    mov  a,m
23C2:    ani  $f0          # only high nibble
23C4:    cpi  $70
23C6:    rnz

23C7:    lxi  h,$43a4      #       v43a4_JT1-indexer
23CA:    mvi  m,$06        #                       := 06
```

```
23CC:    inr  l             #       v43a5_JT1counter
23CD:    mvi  m,$60         #           := 60
23CF:    mvi  l,$63         #       v4363_Spaceship-hit
23D1:    mvi  m,$ff         #           := FF
23D3:    ret


3b49
------------------------------------------------------
23D6:    lxi  h,$43b8       #        v43b8_staging0-F_videobit1
23D9:    mov  a,m
23DA:    ani  $0f           # maintain low-end nibble
23DC:    cpi  $01           #  smallbird fighting stage
23DE:    jz   $3a98

23E1:    cpi  $03           #  smallbird rapid laser fighting stage
23E3:    jz   $3a98
23E6:    cpi  $05           #  bigbird fighting stage
23E8:    jz   $3ad0

23EB:    cpi  $07           # pre-game over stage?
23ED:    jz   $3ad0

23F0:    cpi  $09           # upcoming boss-ship anim stage
23F2:    rc

23F3:    cpi  $0b           # bigboss & smallbird fighting stage

23F5:    jc   $3b02
23F8:       call $3b02
23FB:    jmp  $3a98



JT1:6->040d
------------------------------------------------------ explosion sequence ?
2400:       call $242c
2403:    jz   $2552

2406:    cpi  $20
2408:    jc   $246a
240B:    jz   $2520

240E:    mov  b,a
240F:    rrc
2410:    nop
2411:    mov  a,b
2412:    jnc  $20e8

2415:    mov  a,e
2416:    sui  $05
2418:    adi  $c0
241A:    mov  c,a
241B:    mov  a,d
241C:    aci  $00
241E:    mov  b,a
241F:    mov  a,m
2420:    lxi  d,$2a00       # explosion pattern start
2423:    lxi  h,$2b00       # explosion pattern end
2426:    jmp  $2085



2400
------------------------------------------------------
242C:    lxi  h,$43b9       #   v43b9_videoscroll
242F:    mov  a,m           #
2430:    ani  $f8           #
2432:    mov  m,a           #
2433:    stax $5800         VIDEO Scroll Reg
2436:    lxi  d,$41c6       #   scherm locatie (kolom#11,rij#6)
```

```
2439:    rrc               #
243A:    rrc               #
243B:    rrc               #
243C:    mov  b,a          #
243D:    mov  a,e          #
243E:    sub  b            #
243F:    ani  $1f          #
2441:    mov  b,a          #
2442:    mov  a,e          #
2443:    ani  $e0          #
2445:    ora  b            #
2446:    mov  e,a          #
2447:    mvi  l,$a5        #   v43a5_JT1counter
2449:    dcr  m
244A:    mov  a,m
244B:    ret

JT1:7->040d
------------------------------------------------------
244C:    lxi  h,$43a5      #   v43a5_JT1counter
244F:    dcr  m            #       --1
2450:    mov  a,m          #
2451:    rrc               #
2452:    jc   $06f0        #   scroll background and plot
2455:    ana  a            #
2456:    rnz               #
                           #
2457:    dcr  l            #   v43a4_JT1-indexer
2458:    mvi  m,$02        #                   := 02
245A:    mvi  l,$b8        #    v43b8_staging0-F_videobit1
245C:    mov  a,m          #
245D:    ani  $f0          # only high nibble
245F:    adi  $10          #
2461:    mov  m,a          #
2462:    mvi  l,$ba        #   v43ba_smallbird_fleetsize
2464:    mvi  m,$10        #           := $10
2466:    jmp  $0380        #   schoon foreground behalve HUD (1e 3 regels)


(2408)
------------------------------------------------------
246A:    lxi  b,$0914      #   B=09, C=14    loopvars
246D:    lxi  d,$4ac6      #   background screenpos for action xy(3,6)
2470:    lxi  h,$1c00      #       SCREENdata of Boss-level source-location
2473:    jmp  $0ad6        # (to plot 14x9 bytes)

(26cc)
------------------------------------------------------
2476:    mov  a,b          #
2477:    add  c            #
2478:       call $2495     #
247B:    mvi  l,$d3        #       4bd3
247D:    mov  m,a          #
247E:    lxi  h,$v43bb_remaining_bigbird_count       #
2481:    mvi  a,$08        #       max no. of birbirds
2483:    sub  m            #       8- v43bb_remaining_bigbird_count
2484:    rlc               #
2485:    mvi  l,$9a        #       v439a_16bits_counter_hi
2487:    add  m            #
2488:    rlc               #
2489:    mov  b,a          #
248A:    mvi  l,$6f        #       436f
248C:    mov  a,m          #
248D:    ani  $1e          #               0001.1110
248F:    add  b            #
2490:    stax $4bd1        #       4bd1
2493:    ret

    # 2494: 1F         rar

2478
```

```
------------------------------------------------------
2495:    add  b
2496:    dcr  c
2497:    rz

2498:    add  b
2499:    dcr  c
249A:    rz

249B:    add  b
249C:    dcr  c
249D:    rz

249E:    add  a
249F:    ret

2006
------------------------------------------------------
24A0:    ldax $43b8          #        v43b8_staging0-F_videobit1
24A3:    ani  $0f            # maintain low-end nibble
24A5:    cpi  $08            #       game over staging
24A7:    rc

24A8:    lxi  d,$43c4        #       43c4    (special object slot #1 laser beam)
24AB:    lxi  h,$43e6        #       43e6    (ship laser projectile top ahead)
24AE:       call $2351       #
24B1:    ldax $439b          #       v439ab_16bits_counter_low
24B4:    ani  $03
24B6:    cpi  $03
24B8:    rnz

24B9:    jmp  $24f2          # each 1/3 part of the count we perform

?
------------------------------------------------------
    24BC:       call $2351
    24BF:    ret



2160, 2180, 21c5
------------------------------------------------------
24C4:    ldax $43b8          #        v43b8_staging0-F_videobit1
24C7:    ani  $0f            # maintain low-end nibble
24C9:    cpi  $08            #       game over staging
24CB:    jc   $06f0          #       scroll background and plot

24CE:       call $24e0
24D1:    lxi  h,$43aa        # v43aa_8bits-counter
24D4:    inr  m              #                    ++
24D5:    mov  a,m
24D6:    ani  $03
24D8:    jz   $22fa
24DB:    jmp  $2322          #  anim parts of boss-spaceship?

    # 24DE: 24         inr  h
    # 24DF: BF         cmp  a


24c4
------------------------------------------------------
24E0:    ldax $43aa          #   v43aa_8bits-counter
24E3:    ani  $0f            # maintain low-end nibble
24E5:    rnz

24E6:    ldax $43b9          #       v43b9_videoscroll
24E9:    cpi  $a0            #
24EB:    rc

24EC:    jmp  $067a          #       scroll background and plot starfield
```

```
(24b9)
------------------------------------------------------
24F2:       call $30aa      # fetch cyclic value in range [0-F]
24F5:    adi  $60
24F7:    nop
24F8:    mov  b,a
24F9:    lxi  h,$439b       #       low-val (v439ab_16bits_counter)
24FC:    ani  $0e           # 1101
24FE:    ana  m
24FF:    rnz

2500:    ldax $439e         #  v439e_wing.YY_hiboundary
2503:    cmp  b
2504:    rnc

2505:    ldax $439f         #  v439f_wing.YY_lowboundary
2508:    cmp  b
2509:    rc

250A:    mov  a,b
250B:    sui  $04
250D:    mov  b,a
250E:    ldax $43b9         #       v43b9_videoscroll
2511:    cma                #       complement
2512:    inr  a
2513:    ani  $f8
2515:    adi  $48
2517:    mov  c,a
2518:    push h             # PUSH -> PC   (pop 25DC)
2519:    push h             #              (pop 25DB)
251A:    jmp  $25b7



(240b)
------------------------------------------------------
2520:    push d
2521:       call $0380      #       schoon foreground behalve HUD (1e 3 regels)
2524:    pop  d             #
2525:    ldax $43b9         #       v43b9_videoscroll
2528:    adi  $60           #
252A:    rrc                #
252B:    mov  b,a           #
252C:    ldax $43b8         #        v43b8_staging0-F_videobit1
252F:    ani  $f0           # only high nibble
2531:    add  b
2532:    mvi  b,$90
2534:    jc   $253d         #  convert [B=90] to BCD

2537:    cpi  $90
2539:    jnc  $253d         #  convert [B=>90] to BCD

253C:    mov  b,a           # convert [A] to BCD

(2534)
253D:    xra  a             #       [A] := 0
253E:    mov  a,b
253F:    daa                #       Decimal adjust accumulator
2540:    lxi  h,$435d       #       BCD conversion needed?
2543:    mov  m,a
2544:    inr  l             # 435e
2545:    mvi  m,$00         #      := 00
2547:    mov  a,e
2548:    sui  $5e
254A:    mov  e,a           #
254B:    mvi  b,$04         #       4 karakters
254D:    jmp  $00c4         #   plot(@d,#b)
```

```
(2403)
------------------------------------------------------
2552:   mvi l,$a4
2554:     mvi  m,$07            #  v43a4_JT1-indexer := 07
2556:     inr  l                #
2557:     mvi  m,$40            #   v43a5_JT1counter := 40
2559:     mvi  l,$6b            # 436b
255B:     mvi  m,$ff            #   v436b_8bitscounter := ff
255D:     ret



(2173),2199
------------------------------------------------------ wingdata boundary checks
# 4b70/90 wingformation base
2560:    lxi  h,$4393       #   v4393_JT6_indexer
2563:    mov  a,m
2564:    ani  $01           #   bit #0
2566:    rlc
2567:    rlc
2568:    rlc
2569:    rlc
256A:    rlc                #       bit #0 (* 32)   (0|20)
256B:    adi  $70           #       70 | 90
256D:    mov  l,a           #
256E:    mvi  h,$4b         #       4b70 | 4b90v4357_wing.ZZ_hiboundary
2570:    mvi  e,$08         #  loopcount (08 for all the 8 wingdata objects)
2572:    ldax $4357         #   v4357_wing.ZZ_hiboundary  (..*8+ad)
2575:    rlc
2576:    rlc
2577:    rlc                #       * 8
2578:    nop
2579:    adi  $ad
257B:    mov  d,a
257C:    ldax $439f         # v439f_wing.YY_lowboundary  +3
257F:    adi  $03
2581:    mov  c,a
2582:    ldax $439e         # v439e_wing.YY_hiboundary   -0a
2585:    sui  $0a
2587:    mov  b,a
(2592 e-count) # initial value 08 (2570)
2588:    push h
2589:       call $2596
258C:    pop  h
258D:    mov  a,l
258E:    adi  $04           # HL +4 (next tupple)
2590:    mov  l,a
2591:    dcr  e             # loop countdown
2592:    jnz  $2588
2595:    ret

2589
------------------------------------------------------
# wing WXYZ processing SKIP conditions [W->bit#3=0, X=08,X>=88 Y<b,Y>=c, Z>=d,Z<80]
# ww= wing status #bit 3=?
# xx= char pos offset @1400
# yy= screenpos hi offset (yyyy.---- value)
# zz= screenpos lo offset (zzzz.---- value)
2596:    mov  a,m       # 4b70 wingdata W
2597:    ani  $08       # bit #3 set?    wing.val0 (W)
2599:    rz             # no

259A:    inr  l         # 4b71 | 4b91?   wing.val1 (X)
259B:    mov  a,m
259C:    cpi  $08
259E:    rz

259F:    cpi  $88       #   (1000.1000 bits #3,7)
25A1:    rnc

25A2:    inr  l         # 4b72 | 4b92?   wing.val2 (Y)
```

```
25A3:    mov  a,m
25A4:    cmp  b
25A5:    rc

25A6:    cmp  c
25A7:    rnc

25A8:    inr  l         # 4b73 | 4b93?    wing.val3 (Z)
25A9:    mov  a,m
25AA:    cmp  d
25AB:    rnc

25AC:    cpi  $80
25AE:    rc


25AF:    nop
25B0:    nop
25B1:    nop
25B2:    nop
25B3:    nop
25B4:    mov  c,a
25B5:    dcr  l         # 4b72|4b92? (wing.val2 Y)
25B6:    mov  b,m       # B=4b72/C=4b73 (4b92/4b93)

#[W->bit#3=1 + (X<>08  X<88),(b <= Y < c), (80 >= Z < d]
(251A)
------------------------------------------------------
#   [D-count] =03 @[0-9] =04 @[10-19] =5 @[20-..]
25B7:    ldax $43b8         #        v43b8_staging0-F_videobit1
25BA:    mvi  d,$03
25BC:    cpi  $10           #               (=10 2nd palB?)
25BE:    jc   $25ca         #  (v43b8 < 10 mode)

25C1:    mvi  d,$04
25C3:    cpi  $20
25C5:    jc   $25ca         #  (10 <= v43b8 < 20 )
25C8:    mvi  d,$05         # 5 objects to process
(25be,25be)
25CA:    lxi  h,$43cc       #  (special object slot #3)
(25d8 d-count)
25CD:    mov  a,m
25CE:    ani  $08           # bit #3
25D0:    jz   $25e0         #      =0
# bit #3=1, continue to next special object tupple
25D3:    mov  a,l
25D4:    adi  $04
25D6:    mov  l,a           # 43d0 obj (obj#4) 43d4/d8/dc
25D7:    dcr  d
25D8:    jnz  $25cd
25DB:    pop  h             #       pop opposing 2519 push
25DC:    pop  h             #       pop ''       2518 (sim PC jump)
25DD:    ret

(25d0)
#renew special object tupple (1st value bit#3=0 case)
#   B=wing.val+2, C= wing.val+3 (fe 4b72/4b73|4b92/4b93)
------------------------------------------------------ enemy bombs
#   filling objects 43cc/43d0/43d8/43dc ..
25E0:    mov  a,b
25E1:    adi  $04
25E3:    mov  b,a       #   B= B+4   (0100)
25E4:    mov  a,c
25E5:    adi  $0c
25E7:    mov  c,a       #   C= C+0c
25E8:    mvi  m,$08     # tupple 1st value = 08
25EA:    inr  l
25EB:    mov  a,b
25EC:    rrc
25ED:    ani  $03
25EF:    mov  d,a       #   D=(bits #1/2 from B   <-- B/2 only bit#0/1 )
```

```
25F0:    mov  a,c
25F1:    ani  $04
25F3:    add  d
25F4:    adi  $58       #   A= (C bit #2) + D + 58
25F6:    mov  m,a       # tupple 2nd value = [A]
25F7:    inr  l
25F8:    mov  m,b       # tupple 3rd value = [B]
25F9:    inr  l
25FA:    mov  m,c       # tupple 4rd value = [C]
25FB:    pop  h         #       pop opposing 2519 push
25FC:    pop  h         #       pop ''       2518 (sim PC jump)
25FD:    ret


3406
--------------------------------------------------------
2600:    nop
2601:    nop
2602:    nop
2603:    nop
2604:    nop
2605:    ldax $43b9         #       v43b9_videoscroll
2608:    cma                #       compleement
2609:    rrc
260A:    rrc
260B:    rrc
260C:    ani  $1f
260E:    lxi  h,$4bd2       #  smallbird8_newpos?
2611:    mov  m,a
2612:    inr  l
2613:    ldax $4bd1
2616:    cmp  m
2617:    jc   $2650

261A:    ldax $4bd5
261D:    mov  d,a
261E:    ani  $03
2620:    mov  e,a
2621:    ldax $439b         #       v439ab_16bits_counter_low
2624:    rlc
2625:    rlc
2626:    ani  $0c
2628:    add  e
2629:    adi  $d0
262B:    mov  l,a
262C:    mvi  h,$3e         # 3ed0+x
262E:    mov  a,d
262F:    rrc
2630:    rrc
2631:    ani  $07
2633:    add  m
2634:    mov  d,a
2635:    ldax $43b9         #       v43b9_videoscroll
2638:    sub  d             #
(2662)
2639:    stax $43b9         #       v43b9_videoscroll
263C:    stax $5800         VIDEO Scroll Reg
263F:    ldax $439b         #       v439ab_16bits_counter_low
2642:    rrc
2643:    jnc  $26d0
2646:       call $2668
2649:    jmp  $26aa


(2617)
--------------------------------------------------------
2650:    ldax $439b
2652:    sbb  e
2653:    mov  b,e
2654:    rlc
2655:    rlc
```

```
2656:    ani  $0c
2658:    add  m
2659:    adi  $d0
265B:    mov  l,a
265C:    mvi  h,$3e         # 3ed0+x
265E:    ldax $43b9         #       v43b9_videoscroll
2661:    add  m
2662:    jmp  $2639

    # 2665: D2 AE 26   ??jnc  $26ae??


2646
--------------------------------------------------------
2668:    ldax $436e
266B:    nop
266C:    mov  b,a
266D:    ldax $439a         # v439a_16bits_counter_hi
2670:    cpi  $18
2672:    jc   $2676
2675:    inr  b
(2672)
2676:    cpi  $10
2678:    jc   $267c
267B:    inr  b
267C:    ldax $43ba         #   v43ba_smallbird_fleetsize
267F:    cpi  $03
2681:    jnc  $2685
2684:    inr  b
(2681)
2685:    ldax $4bd6         # smallbird #9 wingpos+2
2688:    adi  $e0
268A:    mov  l,a
268B:    mvi  h,$3e         # 3ee0+x
268D:    mov  a,b
268E:    cmp  m
268F:    jc   $2693
2692:    mov  a,m
(268f)
2693:    mov  d,a
2694:    ldax $v43bb_remaining_bigbird_count           #   v43bb_remaining_bigbird_count
2697:    cpi  $04
2699:    jnc  $269d
269C:    inr  d
(2699)
269D:    cpi  $02
269F:    jnc  $26a3
26A2:    inr  d
(26a3)
26A3:    mov  a,d
26A4:    stax $4bd5
26A7:    ret

    # 26A8: 00         nop
    # 26A9: 58         mov  e,b


(2649)
--------------------------------------------------------
#(4bd3 4bd4 4bd6 4bd7, bytes from smallbird wingpos #8 and #9 ?)
26AA:    lxi  h,$4bd3       #       4bd3
26AD:    mov  a,m
26AE:    dcr  m             #           4bd3--
26AF:    ana  a
26B0:    rnz                # return if 4bd3 <> 0

26B1:    inr  m             #           4bd3++
26B2:    mvi  l,$d6         #       4bd6
26B4:    mov  a,m
26B5:    cpi  $16           #   (20)
26B7:    rnc                # return if 4bd6 >= 20
```

```
26B8:    cpi  $08
26BA:    rc                # return if 4bd6 < 8

26BB:    inr  l            #       4bd7
26BC:    sub  m            #           4bd6 val - 4bd7
26BD:    rlc               #           *2
26BE:    mov  b,a
26BF:    ldax $436f        #       436f   (one of hit collision flag?)
26C2:    ani  $03
26C4:    mvi  l,$d4        #       4bd4
26C6:    mov  m,a          #
26C7:    cma               #   complement
26C8:    ani  $03
26CA:    inr  a
26CB:    mov  c,a
26CC:    jmp  $2476

    # 26CF: C9         ret


(2643)
------------------------------------------------------
26D0:    lxi  h,$4ba8      # enemy unit sprite sequence
26D3:    lxi  b,$0800
26D6:    lxi  d,$8000
(26eb)
26D9:    mov  a,m
26DA:    ana  a
26DB:    jz   $26e5

26DE:    mov  a,d
26DF:    rlc
26E0:    jnc  $26e4
26E3:    mov  d,c
(26e0)
26E4:    mov  e,c
(26d8)
26E5:    inr  c
26E6:    mov  a,l
26E7:    sub  b
26E8:    mov  l,a
26E9:    cpi  $68
26EB:    jnz  $26d9

26EE:    ldax $4bd2        # smallbird8_newpos?
26F1:    add  d
26F2:    add  e
26F3:    ani  $1f
26F5:    stax $4bd6
26F8:    mov  a,e
26F9:    sub  d
26FA:    stax $4bd7
26FD:    ret



0027
------------------------------------------------------ PL1/2 (max/)score handling
2700:    lxi  h,$43a2      #       v43a2_curplayer
2703:    mov  a,m          #
2704:    ana  a            #
2705:    rz                #       v43a2_curplayer > 0 bij PL1/PL2 select
                           #
#  PL1/PL2                  #
2706:    inr  l            #
2707:    mov  a,m          #       which player? v43a3_PLtoggle (value 0 or 1?) offset for 4383/4387
2708:    ani  $01          #  1
270A:    rlc               # *2
270B:    rlc               # *2
270C:    adi  $83          #
```

```
270E:    mov  l,a           #       eihter 4383 (PL1) or 4383+4 (PL2)
270F:    mvi  a,$ff         #
                            #-------
2711:    stax $4397         #       v4397_busyflag := FF
2714:    lxi  d,$4370       #       4370        DEAD-anim slot #0
(2720)
2717:       call $2748      #
271A:    inr  e             #
271B:    inr  e             #
271C:    inr  e             #
271D:    mov  a,e           #           [A]:=74,78,7c
271E:    cpi  $80           #           var 4380 bereikt?  (4380=HiScorePL1)
2720:    jnz  $2717         #        next DE

2723:    mvi  e,$5d         #       DE=435d
2725:    ldax d             #
2726:    ana  a             #
2727:    jz   $2735         #           if 435d = 0 nothing to do -->

272A:    mov  b,a           #
272B:    mvi  c,$00         #
272D:       call $0220      #       BCD conversie?
2730:    xra  a             #           eor a-> [A:=0]
2731:    stax d             #           [D] := 0
2732:    stax $4397         #       v4397_busyflag := 00
                            #-------
(2727)
2735:    ldax $4397         #       v4397_busyflag
2738:    ana  a             #
2739:       cz   $2768      #
273C:       call $27a8      #       SoundAB afspelen (reset to 0F)
273F:    jmp  $3a10

    # 2742: C3 BD 27   ??jmp  $27bd??


2717
------------------------------------------------------- calculate score based on died (enemy only?) subject
2748:    ldax d             #       [A]:= *4370         (4370 = dead-anim slot pointer)
2749:    inr  e             #
274A:    cpi  $01           #
274C:    rnz                # return if value <> 1
  #--------------------- 4370 = 1
274D:    ldax d                 # 4371
274E:    ana  a             #
274F:    rz                     # return when 4371 ==0 (dead-anim slot#0 [01 00 .. ..])
  #-------------------- 4371 <> 0
  # slot #0 example [01 nn .. ..]
2750:    rrc                #     rotating high nibble  to low (4x rrc  nn / 16)  [high low]
2751:    rrc                #     and low to high (because of the carry)          [low high]
2752:    rrc                #
2753:    rrc                #
2754:    mov  b,a           # nibble swap 4371 to [B]
2755:    ani  $f0           # only new high nibble
2757:    mov  c,a           #
2758:    mov  a,b           #
2759:    ani  $0f           # maintain low-end nibble
275B:    mov  b,a           #
275C:       call $0220      # BCD conversion?
275F:    xra  a             # a := eor a
2760:    stax d             # 4371 :=
2761:    stax $4397         # v4397_busyflag
2764:    ret


2739
------------------------------------
2768:    push h
2769:     lxi  d,$4261      #       linkerkant score 000000 player 1
276C:     mvi  b,$06         #
276E:     ldax $43a3            #       v43a3_PLtoggle := [A]
```

```
2771:     ana  a            #
2772:     jz   $2778        #
2775:     lxi  d,$4021      #       PL2score 000000 rechterkant

(2772)
2778:       call $00c4      #       plot
277B:    pop  h
277C:    lxi  d,$43bd
277F:    xchg
2780:    mov  a,m
2781:    inr  l
2782:    ora  m
2783:    rz

2784:    inr  l
2785:    xchg
2786:       call $0314      #       iets met BCD berekenen?
2789:    rnc

278A:    ldax $43a3        #       v43a3_PLtoggle
278D:    adi  $90          #
278F:    mov  l,a          #       PL1/PL2 #lives afhankelijk van waarde palbit0?
2790:    inr  m            #
2791:       call $0367     #       werk nr. of lives bij
2794:    mvi  a,$ff        #
2796:    stax $436a        #       436a := FF
2799:    mvi  l,$be        #       v43be_bonus-lives
279B:    mov  a,m          #       [A] = v43be_bonus-lives
279C:    mvi  m,$00        #       v43be_bonus-lives := 0
279E:    rrc               #
279F:    rrc               #
27A0:    rrc               #
27A1:    rrc               #
27A2:    dcr  l            #       43bd
27A3:    mov  m,a          #           := [A]/16
27A4:    ret



273c
------------------------------------------------------ SoundAB afspelen en reset to 0F
27A8:    lxi  h,$438c      #   v438c_SoundControlA
27AB:    mov  a,m          #
27AC:    stax $6000        Sound Control A  vb $29
27AF:    inr  l            #   v438d_SoundControlB
27B0:    mov  a,m          #
27B1:    stax $6800        Sound Control B  vb $0f
27B4:    ori  $0f          #
27B6:    mov  m,a          #   en weer resetten naar 0F
27B7:    dcr  l            #   voor zowel
27B8:    mvi  m,$0f        #   v438c_SoundControlA en v438d_SoundControlB
27BA:    ret



3b55
------------------------------------------------------
27BD:    lxi  h,$4363      #   v4363_Spaceship-hit /
27C0:    mov  a,m
27C1:    ana  a
27C2:    jnz  $27e2        #   jump if v4363_Spaceship-hit <> 00

27C5:    mvi  l,$61        #   v4361_laser-countdown
27C7:    mov  a,m          #
27C8:    ana  a            #
27C9:    rz                #   return if v4361_laser-countdown := 0

27CA:    cpi  $19          #   #v4361_laser-countdown from 30 till 19 yet?
27CC:    jnc  $27d8        #
                           #                   v4361_laser-countdown =< 18
27CF:    dcr  m            #   v4361_laser-countdown --
```

```
27D0:    mvi  l,$8c         #   v438c_SoundControlA
27D2:    mov  a,m
27D3:    ori  $40
27D5:    mov  m,a
27D6:    ret


(27cc)
------------------------------------------------------
27D8:    mvi  m,$18
27DA:    mvi  l,$8c         #   v438c_SoundControlA
27DC:    mov  a,m
27DD:    ani  $bf
27DF:    mov  m,a
27E0:    ret



(27c2)
------------------------------------------------------ spaceship hit detected?
27e2:    cpi $40            # v4361_laser-countdown
27E4:    jc   $27e9         #   smaller than 40 then jump

27E7:    mvi  m,$40         # setting v4361_laser-countdown to := 40
(27e4)
27E9:    dcr  m             #  v4361_laser-countdown --
27EA:    mvi  l,$8c         #   v438c_SoundControlA
27EC:    mvi  m,$8f         #                        := 8f
27EE:    ret

[40] [3e c8 d8] [e2]

============================================================
    2800:  00 32 00 00 00 00 00 00 00 00 00 00 00 42 42  (<- 207B)
    2810:  00 00 00 00 00 00 00 00 00 00 E1 00 00 E2 00 00
    2820:  32 00 00 00 00 00 00 00 00 E0 00 00 40 00 00 C3
    2830:  00 00 00 00 00 00 DF 00 00 E2 00 00 E0 00 E1 00
    2840:  00 30 00 00 00 00 DE 00 00 00 C2 00 40 00 E0 00
    2850:  00 00 00 30 00 30 00 5A 00 00 E1 00 40 00 E2 00
    2860:  00 00 00 00 00 00 00 30 C1 3E 00 E0 00 40 C2 00
    2870:  00 00 00 00 00 00 00 00 00 5A C1 3E C8 D8 00 00  explosion sequence??
    2880:  E0 E1 C2 E2 E0 00 E1 00 C2 00 E2 CE CA DA 00 00
    2890:  00 00 00 00 00 00 00 00 CF CF C3 3F C2 41 E0 00
    28A0:  00 00 00 00 00 00 00 DE 00 3F 00 C2 41 00 E1 00
    28B0:  00 00 00 00 00 3D DF 3D 00 00 E1 00 41 00 00 C2
    28C0:  00 00 00 3D 00 00 00 00 00 E0 00 00 41 00 00 E2
    28D0:  00 00 3D 00 00 00 00 00 E2 00 00 00 00 4F 00 E0
    28E0:  00 3B 00 00 00 00 00 00 00 C2 00 00 00 4F 00 00
    28F0:  00 00 3B 00 00 00 00 00 00 00 00 00 00 00 4D 4D

    2900:  00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 38  (<- 207E)
    2910:  00 34 00 28 00 00 00 00 00 00 00 00 00 00 00 00
    2920:  00 00 00 00 00 00 00 00 00 00 00 00 10 00 02 00 00
    2930:  00 01 00 00 12 00 00 00 00 00 00 00 00 00 00 00
    2940:  00 00 00 00 00 00 00 00 00 00 10 00 00 80 48 00 04
    2950:  40 08 00 50 00 00 80 10 00 00 00 00 00 00 00 00
    2960:  00 00 00 00 00 00 00 00 10 00 00 20 44 00 00 00 02
    2970:  10 00 00 04 00 48 20 00 00 10 00 00 00 00 00 00
    2980:  00 00 00 00 00 10 00 00 00 44 08 00 00 01 00 00
    2990:  08 00 00 02 00 00 00 84 08 00 00 20 00 00 00 00
    29A0:  00 00 00 20 00 00 00 42 02 00 80 00 00 00 00 00
    29B0:  04 00 00 01 00 00 00 00 00 82 04 00 00 20 00 00
    29C0:  00 40 00 00 01 82 00 00 40 00 00 00 00 00 00 00
    29D0:  02 00 00 00 80 00 00 00 00 00 00 00 81 02 00 00 40
    29E0:  02 80 00 04 00 00 40 00 00 00 00 00 00 00 00 00
    29F0:  01 00 00 00 00 00 40 00 00 00 00 00 00 02 04 08

    2A00:  00 00 00 00 00 00 00 D2 00 00 00 00 00 00 00 00
    2A10:  00 00 00 00 00 DE 00 5E E0 00 00 E1 00 00 00 00
    2A20:  00 00 C1 00 00 CF 53 E2 00 D2 E0 00 00 D0 00 00
    2A30:  00 00 00 DE 00 CE 53 E1 D1 E3 00 E1 D3 00 00 00
```

```
2A40:  00 00 CF C0 DE DF 53 D3 E2 00 E2 D2 00 5E E2 00
2A50:  00 00 00 CE C1 C2 DE D2 E1 E3 D1 00 D2 00 00 00
2A60:  00 00 00 00 DF DE C2 CF E0 D0 E2 E1 C2 C3 00 00
2A70:  DF DE CF CE DF DE CF C8 D8 5E CE 00 CF DE DF CE  explosion sequence??
2A80:  E0 E3 E2 E1 00 E0 D1 CA DA D1 D2 D3 D0 D1 D2 D3
2A90:  00 00 00 00 E3 D2 CE D2 E2 E0 D3 D1 D3 00 00 00
2AA0:  00 00 00 E2 D3 CF DF E1 D0 E3 E1 D2 00 00 00 00
2AB0:  00 00 E1 D0 DE 00 DE E2 00 D3 53 E2 5E C1 C0 00
2AC0:  00 00 00 DF 00 00 CF 5E D1 D2 00 53 E3 00 00 00
2AD0:  00 00 CE 00 CF 00 CE D2 D2 00 53 00 5E E0 00 00
2AE0:  00 00 00 00 00 DE 00 E1 D3 00 E2 00 00 00 00 00
2AF0:  00 00 00 00 00 00 00 5E D0 00 00 00 00 00 00 00

2B00:  00 00 00 00 00 00 00 00 00 00 80 01 40 02 80 05
2B10:  A0 01 40 02 00 01 00 00 00 00 00 00 00 00 00 00
2B20:  00 00 00 00 00 00 80 00 00 01 20 04 00 01 40 12
2B30:  48 02 80 01 20 04 00 00 00 01 00 00 00 00 00 00
2B40:  00 00 00 00 80 00 00 02 10 08 00 01 80 04 A0 21
2B50:  84 05 20 02 80 01 10 08 00 00 00 01 00 00 00 00
2B60:  00 00 80 00 00 04 08 10 00 01 40 00 40 0A 10 40
2B70:  02 08 40 00 10 04 80 02 08 10 00 00 00 01 00 00
2B80:  80 00 00 08 04 20 00 02 20 00 20 14 00 01 08 80
2B90:  01 10 80 02 20 00 08 04 80 02 04 20 00 00 00 01
2BA0:  01 01 01 01 01 04 20 00 10 28 80 02 04 00 00 04
2BB0:  20 20 00 04 40 01 10 00 04 08 80 04 00 00 00 00
2BC0:  00 00 00 08 20 00 88 10 00 44 00 00 00 10 02 00
2BD0:  08 40 00 00 00 08 40 00 08 01 00 10 80 04 00 00
2BE0:  00 00 20 00 84 20 00 08 00 00 00 00 00 20 01 00
2BF0:  04 80 00 00 00 00 00 10 40 00 04 01 00 00 80 00

2C00:  [0B 0C 0D 0E 0B 0C 0A 0A 0A 0A 0A 0A 0A 06 06 1E   <--- smallbird  pattern @2c00
2C10:  03 03 1F 05 05 1C 04 04 04 1D 06 06 1A 04 04 04
2C20:  1B 05 05 05 05 18 1F 07 07 07 07 07 07 07 07 07
2C30:  00] FF FF FF [05 05 1C 04 1D 0A 0A 0A 0A 0A 0A 06  <--- smallbird  pattern @2c34
2C40:  06 1E 03 03 1F 05 1C 04 04 1D 0A 06 06 1E 03 03
2C50:  1F 05 1C 04 04 1D 0A 06 06 1E 03 03 1F 05 1C 04
2C60:  04 1D 0A 06 1E 03 1F 05 1C 04 1D 06 1E 03 03 03
2C70:  03 15 16 17 01 01 05 05 01 01 05 05 01 01 05 05
2C80:  01 01 05 05 02 02 18 07 07 07 00] FF FF FF FF FF
2C90:  [1C 04 04 04 04 04 04 04 04 04 04 04 04 04 04 1D  <--- smallbird  pattern @2c90
2CA0:  06 06 06 06 06 06 06 1E 03 03 03 03 03 03 1F 05
2CB0:  05 05 05 1C 04 04 1D 06 09 09 09 1E 03 07 07 08
2CC0:  08 07 07 08 07 00] FF FF [05 05 05 05 1C 04 04 04  <--- smallbird  pattern @2cc8
2CD0:  04 04 04 04 04 04 04 04 04 04 04 1D 09 09 09 09
2CE0:  0A 0A 0A 09 0A 0A 06 1E 03 03 03 1F 05 05 18 03
2CF0:  19 06 06 1E 03 03 1F 05 05 05 05 05 05 05 05 00] FF

2D00:  [0B 0C 0D 0E 0B 0C 06 1E 03 03 03 03 03 03 03 03   <--- smallbird  pattern @2d00
2D10:  03 03 03 03 03 03 1F 05 05 1C 04 04 04 04 04 04
2D20:  04 04 04 04 1D 06 06 1E 03 03 03 03 03 03 1F 05
2D30:  05 05 05 05 1C 04 04 04 04 04 04 04 04 04 04 1B
2D40:  00] FF FF FF [05 05 05 18 03 03 03 03 03 03 03 03  <--- smallbird  pattern @2d44
2D50:  03 19 06 06 1A 04 04 1B 05 05 18 03 03 03 03 03
2D60:  03 03 19 06 06 06 06 06 06 06 06 06 06 1A 04 04
2D70:  1B 05 05 1C 04 04 1D 06 06 1A 04 04 1B 05 05 05
2D80:  05 05 05 05 00] FF FF FF [1C 04 04 1D 06 06 09 0A  <--- smallbird  pattern @2d88
2D90:  0A 09 09 09 16 17 14 03 03 03 1F 05 05 1C 04 04
2DA0:  1D 06 06 1E 03 03 03 03 07 07 08 08 07 07 05 05
2DB0:  1C 04 04 04 04 04 04 04 1D 1A 04 1B 00] FF FF FF
2DC0:  [14 03 03 19 06 0A 0A 09 09 09 0A 12 13 10 11 12   <--- smallbird  pattern @2dc0
2DD0:  13 10 11 12 13 10 04 04 04 04 1B 05 18 03 19 06
2DE0:  1A 04 1B 05 18 07 07 07 08 08 07 07 07 03 03 19
2DF0:  0D 0E 00] FF FF FF FF FF FF FF FF FF FF FF FF FF

2E00:  [0B 0C 0D 0E 02 02 02 02 0B 0C 0D 0E 01 01 14 15  <--- smallbird diving pattern bigboss fight level  @2e00
2E10:  16 17 01 01 05 05 05 05 02 02 02 02 00] FF FF FF
2E20:  [0B 0C 0D 0E 0B 0C 0D 0E 02 02 02 02 02 02 02 02  <--- smallbird  pattern @2e20
2E30:  05 05 01 05 05 01 05 05 01 05 05 01 00] FF FF FF
2E40:  [0B 0C 0D 0E 01 01 01 18 03 19 06 06 1A 04 1B 05  <--- smallbird  pattern @2e40
2E50:  18 03 19 06 06 1A 04 04 04 04 04 04 04 04 04 1B
2E60:  05 05 05 01 01 01 01 01 00] FF FF FF [0B 0C 0D 0E  <--- smallbird  pattern @2e6c
```

```
    2E70:  01 01 0B 0C 0D 0E 01 01 05 05 05 05 01 01 0B 0C
    2E80:  0D 0E 01 01 07 08 08 07 08 08 08 07 00] FF FF FF
    2E90:  [14 15 16 17 14 15 16 17 14 03 03 03 03 03 03 03   <--- smallbird  pattern @2e90
    2EA0:  03 03 03 03 03 19 09 0A 0A 09 09 0A 0A 12 13 08
    2EB0:  08 07 07 08 08 08 08 04 04 04 11 12 13 10 11 12
    2EC0:  13 00] FF FF [10 11 12 13 10 11 12 13 10 04 04 04   <--- smallbird  pattern @2ec4
    2ED0:  04 04 04 04 04 04 0A 0A 0A 09 0A 09 0A 09 16 17
    2EE0:  14 03 03 03 07 07 07 07 03 19 06 1A 04 1B 05 18
    2EF0:  07 07 07 07 00] FF FF FF FF FF FF FF FF FF FF FF

    2F00:  [05 1C 04 1D 06 06 06 06 06 09 09 09 0A 0A 0A 09   <--- smallbird  pattern @2f00
    2F10:  09 16 17 14 1F 05 18 03 19 06 1E 03 1F 05 18 03
    2F20:  19 06 1E 03 1F 05 05 1C 08 08 08 08 08 08 08 08
    2F30:  00] FF FF FF [05 18 03 19 06 06 06 06 0A 0A 09 09  <--- smallbird  pattern @2f34
    2F40:  0A 0A 09 0A 0A 12 13 10 1B 05 1C 04 1D 1E 1F 1C
    2F50:  04 1D 06 1A 04 04 1B 05 18 07 07 07 07 08 07 07
    2F60:  07 07 00 FF 0B 0C 0D 0E 0B 0C 1E 03 19 06 1E 03
    2F70:  19 06 1E 03 19 06 1E 1F 1C 1D 1E 03 03 03 1F 05
    2F80:  18 03 19 06 1E 03 1F 05 08 08 08 08 08 08 08 07
    2F90:  07 08 08 08 08 08 00] FF FF FF FF FF FF FF FF FF
    2FA0:  [05 05 18 03 03 03 03 03 03 03 03 19 06 06 06 06  <--- smallbird  pattern @2fa0
    2FB0:  06 06 06 1A 04 1B 05 18 03 03 03 03 19 06 06 06
    2FC0:  1A 04 1B 05 18 03 03 03 03 19 06 06 06 1A 04 1B
    2FD0:  05 18 03 03 03 03 19 06 06 06 1A 04 1B 05 18 03
    2FE0:  03 19 06 06 1A 11 12 13 02 02 02 05 05 02 02 02
    2FF0:  05 05 02 02 02 05 1C 08 08 07 07 08 08 08 00] FF

2153, 2193
================================================== JUMP TABLE 6.  ============================
3000:    lxi  h,$4393       #   v4393_JT6_indexer
3003:    mov  a,m
3004:    inr  m
3005:    ani  $07           # 8 enties max
3007:    lxi  h,$3018       # jump base
300A:    rlc                # *2 (16-bits index)
300B:    add  l
300C:    mov  l,a
300D:    mov  a,m
300E:    inx  h
300F:    mov  l,m       # hi-byte jump table 6
3010:    mov  h,a       # low-byte jump table 6
3011:    pchl           # JT6 JUMP!  (based on HL)

JT6:7 <- 3011
-----------------------------
3012:      ret


"
    [Jump table 6]
    (JT6-index bit#10=0x)   0 -> 3264    2 -> 30BA      4 -> 315a       6 -> 322c
    (JT6-index bit#10=1x)   1 -> 3028    3 -> 3124      5 -> 31b4       7 -> 3012


    3018:  3264 3028 30BA 3124  (base location 3018 see 3007)
    3020:  315A 31B4 322C
    3026:  3012


    Trace of subroutines:
        3264
        3028 -> 305c               -> 3074 -> 30aa
        30BA -> 30da + 30e4 -> 3112 + 3074 -> 30aa
        3124                           -> 30aa
        315a -> 3192
        31b4 -> 3210                    + 30aa
        322c
        3012

    "
```

```
JT6:1 <- 3011
--------------------------------------button afhandeling?
3028:    lxi  h,$4357       # v4357_wing.ZZ_hiboundary
302B:    mov  a,m
302C:    cpi  $03
302E:    rnc                #   return if 4357 >= 03  (wxyZ Z hi-boundary)

302F:    mvi  l,$50         # 4350; vars range
3031:    mov  a,m
3032:    cpi  $04
3034:    rnc                #   return if 4350 >= 04

# v4350 < 4 mode
3035:    mvi  l,$58
3037:    mov  a,m
3038:    ana  a
3039:    jz   $305c         #   return if 4358 == 0

303C:    dcr  m             #   4358--
303D:    rnz                #   return if 4358 <> 0

#  4358 ==00 is countdown for reset action?
#                           4350: 04 2e 00 10 50 .. .. ++ --
303E:    dcr  l             #
303F:    inr  m             #   4357++ (v4357_wing.ZZ_hiboundary)
3040:    mvi  l,$50
3042:    mvi  m,$04         #   4350 := 04
3044:    mvi  l,$53
3046:    mvi  m,$10         #   4353 := 10  10xx as offset for [4b50-4b6f] vars
3048:    inr  l
3049:    mvi  m,$50         #   4354 := 50
304B:    mvi  l,$51
304D:    mvi  m,$2e         #   4351 := 2e  2exx as offset for [4b50-4b6f] vars, see 329B
304F:    inr  l
3050:    mvi  m,$00         #   4352 := 00
3052:    ldax $43c2         #       v43c2_SpaceShip-horizPixelpos   (spec.obj0.val2)
3055:    rrc
3056:    rc                 # return if bit#0 van v43c2_SpaceShip-horizPixelpos == 1

3057:    mvi  m,$40         # 4352 := 40
3059:    ret


(3039)
---------------------------------------- set new v4358 counter value
305C:       call $3074          # [C :=
305F:    lxi  h,$4357       # v4357_wing.ZZ_hiboundary
3062:    mov  a,m
3063:    rlc                    # *2
3064:    rlc                    # *2
3065:    nop
3066:    nop
3067:    add  c
3068:    adi  $07
306A:    mvi  l,$58         # 4358
306C:    mov  m,a           #     := [A]
306D:    ret


3074: 305c, 30e4
------------------------------------------------------ [C] output
3074:    lxi  h,$43b8       #        v43b8_staging0-F_videobit1
3077:    mov  a,m
3078:    rrc                #   01->80, 02->01
3079:    nop
307A:    ani  $07
307C:    mov  b,a
307D:    mvi  a,$07
307F:    sub  b
3080:    mov  c,a
3081:    mov  a,m           #        v43b8_staging0-F_videobit1
```

```
3082:    cpi  $80
3084:    jc   $3089         # als kleiner dan 80

3087:    mvi  a,$70         # 0111.0000 when >= 80
(3084)
3089:    rrc                #   /2
308A:    rrc                #   /2
308B:    rrc                #   /2
308C:    rrc                #   /2 (nibble  xy ->  0x)
308D:    ani  $07
308F:    mov  b,a
3090:    mvi  a,$07
3092:    sub  b
3093:    add  c
3094:    mov  c,a
3095:    ldax $43ba         #   v43ba_smallbird_fleetsize
3098:    sui  $05           #
309A:    jnc  $309f         #   nog minimaal 5 over?

309D:    mvi  a,$10         #   zo nee maak er de default 10 van
(309a)
309F:    add  c             #      *C erbij optellen
30A0:    mov  c,a
30A1:       call $30aa      # fetch cyclic value in range [0-F]
30A4:    ani  $07           #  0000.0xxx
30A6:    add  c
30A7:    mov  c,a
30A8:    ret



24f2, 30a1, 3141, 3161, 31f8, 3560
------------------------------------------------------ fetch cyclic value in range [0-F]
#[A] contains modulo 16 value
30AA:    lxi  h,$439b       #   v439ab_16bits_counter_low
30AD:    mov  a,m           #       low byte value
30AE:    rlc                #       xxxx.xxxy
30AF:    rlc                #       xxxx.xxyy
30B0:    rlc                #       xxxx.xyyy
30B1:    ani  $07           #       0000.0yyy (per 8 cycles)
30B3:    mvi  l,$c2         #   v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
30B5:    add  m             #
30B6:    ani  $0f           #   modulo 16  0000.zzzz  lower nibble
30B8:    ret

    # 30B9: C0         rnz


JT6:2 <- 3011
------------------------------------------------------
30BA:    lxi  h,$4358       #   4358
30BD:       call $30da      # count down 4358 +1     (4359)
30C0:       call $30da      # count down 4358 +1+1   (435a)
30C3:       call $30da      # count down 4358 +1+1+1 (435b)
30C6:    mvi  l,$50         #   4350
30C8:    mov  a,m           #
30C9:    ana  a             #
30CA:    rnz                #  v4350 <> 0

#  v4350 == 0 mode
30CB:    mvi  l,$55         #   4355
30CD:    mov  a,m           #
30CE:    ana  a             #
30CF:    jz   $30e4         #
                            #
30D2:    dcr  m             #   4355 := --1
30D3:    rnz                #

30D4:    mvi  l,$50         # 4350
30D6:    mvi  m,$01         #   4350 := 01
30D8:    ret
```

```
    # 30D9: FE 2C      cpi  $2c


30bd, 30c0, 30c3
------------------------------------------------------ count down m(HL+1) until zero
30da:     inr l            # (4359/435a/435b)
30DB:      mov  a,m
30DC:      ana  a
30DD:     rz               # retun if [m]'s value is zero
  #if one of the above vars (4359/435a/435b) is nonzero
30DE:     dcr  m               # count down m(HL) -1
30DF:     ret

    # 30E0: 7E         mov  a,m
    # 30E1: FE 01      cpi  $01
    # 30E3: D0         rnc

(30cf)
------------------------------------------------------
30E4:       call $3074      # [C := ..]
30E7:    lxi  h,$439a       #V439a_16bits_counter_hi
30EA:    mov  a,m
30EB:    cpi  $10
30ED:    jc   $30f2
#V439a_16bits_counter_hi < $10
30F0:    mvi  a,$0f
(30ed)
30F2:    mov  b,a           # b :=  min( $f, V439a_16bits_counter_hi)
30F3:    mvi  a,$0f
30F5:    sub  b
30F6:    add  c
30F7:    mov  c,a           # [C := $f - B + C]
30F8:    mvi  b,$01
30FA:    mvi  l,$58         # 4358
30FC:       call $3112
[]
30FF:       call $3112
3102:       call $3112
3105:    mov  a,c
3106:    rrc                # / 2
3107:    rrc                # / 4
3108:    ani  $3f           # clear out the possible carry values
310A:    adi  $01
310C:    mvi  l,$55         # 4355
310E:    mov  m,a           #      := c/4 +1
310F:    ret

    # 3110: 21 50 2C   lxi  h,$2c50

30fc, 30ff, 3102
------------------------------------------------------
# initial HL == 4358
3112:   inr l           #call #1 (@30fc): 4359, call #2 (@30ff): 435a, call #3 (@3102): 435b
3113:    mov  a,m
3114:    ana  a
3115:    rnz            # current countdown not zero at either 4359, 435a, 435b?
# counter == 0 mode
3116:    mov  a,c
3117:    rrc            # /2
3118:    ani  $7f       # clear out possible carry value
311A:    mov  c,a
311B:    mov  a,b
311C:    ana  a
311D:    rz             # b == 0?

311E:    dcr  b         # otherwise b-- (to again call 3112 as seen in code above)
311F:    mvi  m,$0c     # reset current countdown to $0c
3121:    ret
```

```
JT6:3 <- 3011
------------------------------------------------------
3124:    lxi  h,$4350
3127:    mov  a,m
3128:    cpi  $01
312A:    rnz               # return if 4350 <> 01

#--- 4350 ==1 mode ----
312B:    mvi  m,$02        # 4350 := 02
312D:    mvi  l,$b8        #  v43b8_staging0-F_videobit1
312F:    mov  a,m          #
3130:    rrc               #
3131:    rrc               #
3132:    ani  $0f          # maintain low-end nibble
3134:    adi  $05          #
3136:    cpi  $11          #
3138:    jc   $313d        #

313B:    mvi  a,$05        #
(3138)                     #
313D:    mvi  l,$57        # v4357_wing.ZZ_hiboundary
313F:    sub  m            #
3140:    mov  b,a          #
3141:       call $30aa     # fetch cyclic value in range [0-F]
3144:    inr  a            #
3145:    cmp  b            #
3146:    jc   $314b        #

3149:    mvi  a,$01        #
(3146)
314B:    mvi  l,$53        # 4353 := 01
314D:    mov  m,a
314E:    ret



JT6:4 <- 3011
------------------------------------------------------
315A:    lxi  h,$4350
315D:    mov  a,m
315E:    cpi  $02
3160:    rnz
#----------------------------- 4350 =2 mode
3161:       call $30aa     # fetch cyclic value in range [0-F]
3164:    nop
3165:    mov  b,a
3166:    rlc
3167:    adi  $50
3169:    mov  l,a
316A:    mvi  h,$4b        #   HL:= 4b50+[0-F]*2 entry   SoundAB
316C:    mov  a,b          #
316D:    rlc               #
316E:    rlc               #
316F:    adi  $70          #
3171:    mov  e,a          #
3172:    mvi  d,$4b        #   DE:= 4b70+[0-F]*4 entry   wingformation-data
3174:    mvi  c,$10        #
3176:    mov  a,c          #
3177:    sub  b            #
3178:    mov  b,a          # b:=$10-b(=0-f) := ..

(318b) #outer-loop C (10-0)  16x
3179:       call $3192     #
317C:    inx  d            #
317D:    inx  d            #
317E:    inx  d            #
317F:    inx  d            #    next DE pointer-val (skip 4-byte tupple)
3180:    inx  h            #
3181:    inx  h            #    next HL pointer-val (16 bits)
3182:    dcr  b            #
```

```
3183:    jnz  $318a         #
3186:    mvi  e,$70         #  DE=4b70??
3188:    mvi  l,$50         #       v4b50_smallbird-romlists
(3183) #inner-loop B
318A:    dcr  c
318B:    jnz  $3179
318E:    ret



3179
------------------------------------------------------
3192:    ldax d
3193:    ani  $08
3195:    rz

3196:    ldax $4394         # v4394_4b50copy
3199:    cmp  m
319A:    rnz

319B:    ldax $4356         # [A := *4356]
319E:    inr  l             # 4357
319F:    mov  b,m
31A0:    dcr  l             # 4356
31A1:    cmp  b
31A2:    rnz                # return if 4356 <> 4357

# 4356 == 4357
31A3:    mov  a,l           #56
31A4:    stax $4354         # 4354 := 56
31A7:    mvi  a,$03
31A9:    stax $4350         # 4350 := 03
31AC:    pop  h
31AD:    ret



JT6:5  <- 3011
------------------------------------------------------
31B4:    ldax $4350
31B7:    cpi  $03
31B9:    rnz

#4350 ==3 mode -----
31BA:    ldax $4354         # 4354 == 50 (@3049), 4354 == 56 (@31A4)
31BD:    sui  $50
31BF:    rlc
31C0:    adi  $72
31C2:    mov  l,a
31C3:    mvi  h,$4b         #       $4b72 + xx verwijzing  (wingformation)
31C5:    mov  b,m           #
31C6:    inr  l             #
31C7:    mov  d,m           #
31C8:    ldax $43c2         #       v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
31CB:    mvi  c,$04
31CD:    cmp  b
31CE:    jnc  $31d6

31D1:    mov  c,a
31D2:    mov  a,b
31D3:    mov  b,c
31D4:    mvi  c,$00
(31ce)
31D6:    sub  b
31D7:    rlc
31D8:    rlc
31D9:    rlc
31DA:    ani  $07           #0000.0xxx
31DC:    adi  $00
31DE:    mov  l,a
31DF:    mvi  h,$33         # 33xx
```

```
31E1:    mov  a,m
31E2:    add  c
31E3:    rlc
31E4:    rlc
31E5:    mov  c,a
31E6:    nop
31E7:    nop
31E8:    nop
31E9:    ldax $4357          # v4357_wing.ZZ_hiboundary
31EC:    mov  b,a
31ED:       call $3210       # determine B based on $4353==1 and value in D (<58=0,<78=1,<98=2,>=98=3)
31F0:    mov  a,c
31F1:    add  b
31F2:    adi  $10
31F4:    mov  l,a
31F5:    mvi  h,$33          # 3310 + b + c
31F7:    mov  c,m
31F8:       call $30aa       # fetch cyclic value in range [0-F]
31FB:    ani  $06
31FD:    add  c
31FE:    mov  l,a
31FF:    mvi  h,$33          # 33xx
3201:    mov  a,m
3202:    inr  l
3203:    mov  b,m
3204:    lxi  h,$4350
3207:    mvi  m,$05          # 4350 := 05
3209:    inr  l
320A:    mov  m,a
320B:    inr  l
320C:    mov  m,b
320D:    ret

    3310:  88 90 98 A0 68 70 78 80 48 50 58 60 48 30 38 40
    3320:  88 90 98 A0 A8 B0 B8 C0 C8 D0 D8 E0 C8 E8 F0 F8


31ed
------------------------------------------------------
3210:    ldax $4353
3213:    cpi  $01
3215:    rnz                 # 4353 <> 1?

#4353==1 mode -----
3216:    mov  a,d            # (50/48/..)
3217:    mvi  b,$00
3219:    cpi  $58
321B:    rc                  # <58?

321C:    mvi  b,$01
321E:    cpi  $78
3220:    rc                  # <78?

3221:    mvi  b,$02
3223:    cpi  $98
3225:    rc                  # <98?

3226:    mvi  b,$03
3228:    ret


JT6:6 <- 3011
------------------------------------------------------
322c:   ldax $4350
322F:    cpi  $04
3231:    rnz

# v4350 ==04 mode
3232:    lxi  h,$4b50        #       v4b50_smallbird-romlists
3235:    lxi  d,$4b70        #       v4b70_wingformation-base
3238:    ldax $4356          #       4356
```

```
323B:    mov  c,a
323C:    ldax $4394         #       v4394_4b50copy
323F:    mov  b,a

(3256 loop e until b0)
3240:    ldax d
3241:    ani  $08           # v4b70_wingformation-base bit #3 set?
3243:    jz   $324e         # yes

3246:    mov  a,m           # fetch v4b50_smallbird-romlists
3247:    cmp  b
3248:    rnz                # return if it is <> 0

3249:    inr  l             # 4b51
324A:    mov  a,m
324B:    dcr  l             # 4b50
324C:    cmp  c
324D:    rnz                # 4b50 <> 4b51 ?

(324e)
#bit #3 = 0 for v4b70_wingformation-base, 4b50's value == 4b51's value
324E:    inr  l             # 4b51
324F:    inr  l             # 4b52
3250:    mov  a,e
3251:    adi  $04
3253:    mov  e,a           #  e += 4
3254:    cpi  $b0           # reaching e == (4b)b0 ?
3256:    jnz  $3240

# e == b0
3259:    mvi  a,$06
325B:    stax $4350         # 4350 := 06
325E:    ret



JT6:0 <- 3011
------------------------------------------------------ set sequence[4b50-4b6f] for smallbirds
3264:     lxi h,$4395       #  4395
3267:    mov  a,m
3268:    stax $4356         #  make copy, 4356 := 4395
326B:    inr  a
326C:    ani  $0f           # maintain low-end nibble
326E:    mov  m,a
326F:    mvi  l,$50         #   4350
3271:    mov  a,m           #
3272:    cpi  $05           #
3274:    rc                 #

#v4350 > 5
3275:    mvi  m,$00         #   4350:=00
3277:    mvi  l,$53         #   4353
3279:    mov  c,m           #
327A:    inr  l             #    4354 (? == 50 / 56)
327B:    mov  l,m           #
327C:    mvi  h,$4b         #   $4bxx  (4b50, 4b56 )?
327E:    ldax $4356         #  4356
3281:    mov  d,a
3282:    ldax $4394         #  v4394_4b50copy   ($10?)
3285:    mov  e,a
3286:    mov  a,l
3287:    sui  $50
3289:    rrc
328A:    mov  b,a
328B:    mvi  a,$10
328D:    sub  b
328E:    mov  b,a

(32ac outer-c loop)
328F:    mov  a,m
3290:    inr  l
```

```
3291:    cmp  e
3292:    jnz  $32a4     # skip setting pointer

3295:    mov  a,m
3296:    cmp  d
3297:    jnz  $32a4     # skip setting pointer

#setting pointer
329A:    dcr  l
329B:    ldax $4351         # [A := 4351 ]
329E:    mov  m,a           #      (4b56/58/5a/5c/ ..5e?)   gets value pointed to by 4351
329F:    inr  l
32A0:    ldax $4352         # [A := 4352 ]
32A3:    mov  m,a           #      (57?/59?/ .. 4b5f) gets value pointed to by 4352

# loop control
(3292,3297)                          # enemy unit flies out of formation?
32A4:    inr  l
32A5:    dcr  b
32A6:    jnz  $32ab
32A9:    mvi  l,$50         #       4350/4b50?
(32a6)
32AB:    dcr  c
32AC:    jnz  $328f
32AF:    ret
========================================== end of JUMP TABLE 6.  ============================



(052f)
-------------------------------------------------------clear & init vars
32B0:    lxi  h,$4350
32B3:    mvi  b,$30
32B5:       call $05d8      #       CLEAR memory [4350 .. 4380]
32B8:    mvi  l,$9a         #       439a
32BA:    mvi  b,$04         #
32BC:       call $05d8      #       CLEAR memory [439a .. 439e]
32BF:    ldax $v43bb_remaining_bigbird_count         #      v43bb_remaining_bigbird_count
32C2:    ana  a             #
32C3:    rz                 #       return bij v43bb_remaining_bigbird_count = 0
--------------------
32C4:    rlc
32C5:    rlc
32C6:    rlc
32C7:    mov  c,a
32C8:    lxi  h,$4b70       #       v4b70_wingformation-base
32CB:    mvi  b,$40         #
32CD:       call $05d8      #       CLEAR memory [4b70 .. 4bb0] wingformation-data
32D0:    mvi  d,$4b         #       4bxx
32D2:    mvi  h,$3f         #       3fxx
32D4:    mvi  a,$40         #
32D6:    sub  c             #
32D7:    adi  $70           #
32D9:    mov  e,a           #
32DA:    adi  $10           #
32DC:    mov  l,a           #
32DD:    mov  b,c           #
32DE:    ldax $43b8         #            v43b8_staging0-F_videobit1
32E1:    rrc                #
32E2:    rrc                #
32E3:    jnc  $05e0         #       copy [B] bytes van rom[HL] naar var[DE]
                            #
32E6:    mov  a,l           #
32E7:    adi  $40           #
32E9:    mov  l,a           #
32EA:    jmp  $05e0         #       copy [B] bytes van rom[HL] naar var[DE]
--------------------------------------

32ED:       call $05e0      #       copy [B] bytes van rom[HL] naar var[DE]
32F0:    jmp  $03a0         #       schoon background met 00
```

```
------------------------------------------------------
"HEXDATA
   accessed by JT6:5(@31B4) from @31DF (block A) and @31FF (block B)
   precondition @31BA is when v4350 ==3
  "

#block A
    3300:  [00 01 02 02 03 03 03 03] FF FF FF FF FF FF FF FF     # 8 main entries to 4 list sets
    # values in list acts as index 3300 + value, 88 -> ..88
    3310:  [88 90 98 A0 68 70 78 80] [48 50 58 60 48 30 38 40]   #listbase 00, listbase 01
    3320:  [88 90 98 A0 A8 B0 B8 C0] [C8 D0 D8 E0 C8 E8 F0 F8]   #listbase 02, listbase 03

#block B
#the following rom-pointers will be put into the main processing pointer [4351 4352] entries for smallbird wave patterns
    3330:  [11 30] [2C 00] [2F A0] [2C 00] - [2E C4] [2F A0] [2F 34] [2F A0]  # ..30, ..38
    3340:  [2C C8] [2E C4] [2E 20] [2E C4] - [11 30] [13 9C] [13 D0] [2C 00]  # ..40, ..48

    3350:  [11 30] [13 28] [2C 00] [2F 34] - [11 A4] [2C 90] [2F 34] [2F A0]  # ..50, ..58
    3360:  [2C 90] [2C C8] [2E 20] [2E C4] - [11 60] [13 54] [13 9C] [13 D0]  # ..60, ..68

    3370:  [10 20] [10 64] [11 A4] [13 28] - [10 20] [11 A4] [12 00] [2F 34]  # ..70, ..78
    3380:  [2C 90] [2C C8] [2D C0] [2E 20] - [11 60] [12 44] [12 88] [13 54]  # ..80, ..88

    3390:  [10 20] [10 64] [12 00] [12 44] - [10 20] [12 00] [10 20] [12 00]  # ..90, ..98
    33A0:  [10 A8] [2D 88] [10 A8] [2D C0] - [11 D0] [12 CA] [13 00] [13 54]  # ..a0, ..a8

    33B0:  [10 20] [10 64] [10 D4] [13 00] - [10 20] [10 D4] [12 00] [2F 00]  # ..b0, ..b8
    33C0:  [2D 00] [2D 44] [2D 88] [2E 6C] - [11 00] [11 D0] [12 CA] [2F 64]  # ..c0, ..c8

    33D0:  [11 00] [13 00] [2F 64] [2F 00] - [10 D4] [2D 00] [2F 00] [2C 34]  # ..d0, ..d8
    33E0:  [2D 00] [2D 44] [2E 6C] [2E 90] - [11 00] [2C 34] [2F 64] [2F 64]  # ..e0, ..e8

    33F0:  [2E 90] [2F 00] [2C 34] [2C 34] - [2D 44] [2E 6C] [2E 90] [2E 90]  # ..f0, ..f8
    ------------------------------------------------------



JT4:{0a,0e}<-080F   bigbird fight - enemy wave sequence-step?
------------------------------------------------------
3400:       call $0876
3403:       call $3800
3406:       call $2600
3409:       call $3800
340C:       call $3980
340F:    ldax $v43bb_remaining_bigbird_count               #   v43bb_remaining_bigbird_count
3412:    ana  a
3413:    jz   $3462             # no remaining bigbirds left

3416:    cpi  $04
3418:    jnc  $3438             # at least 4 bigbirds left

341B:       call $3474
341E:       call $3486
3421:       call $3560
3424:       call $3498
3427:       call $34aa
342A:    ldax $439b             #   v439ab_16bits_counter_low
342D:    rrc
342E:    jc   $0fc0             # bit #0 in carry is 1

3431:       call $3930
3434:    jmp  $0c40


(3418)
------------------------------------------------------
3438:    ldax $439b             #   v439ab_16bits_counter_low
343B:    rrc
343C:    jc   $3452
```

```
343F:       call $3474
3442:       call $3560
3445:       call $3498
3448:       call $3930
344B:    jmp  $0c40


(343c)
------------------------------------------------------
3452:       call $3486
3455:       call $3560
3458:       call $34aa
345B:    jmp  $0fc0


(3413)
------------------------------------------------------
3462:    ldax $439b             #   v439ab_16bits_counter_low
3465:    rrc                    #
3466:    rc                     # bit #0 set (odd number countervalue)
# even counter cycle
3467:       call $0c40
346A:       call $0fc0
346D:    jmp  $2204



341b, 343f
------------------------------------------------------
3474:    lxi  h,$4b70           #   v4b70_wingformation-base
(3482)
3477:    push h
3478:       call $34c0
347B:    pop  h
347C:    mov  a,l
347D:    adi  $08
347F:    mov  l,a
3480:    cpi  $90
3482:    jnz  $3477
3485:    ret


341e, 3452
------------------------------------------------------
3486:    lxi  h,$4b90              #wingdata
(3489)
3489:    push h
348A:       call $34c0
348D:    pop  h
348E:    mov  a,l
348F:    adi  $08
3491:    mov  l,a
3492:    cpi  $b0
3494:    jnz  $3489             # until 4bb0
3497:    ret


3424, 3445
------------------------------------------------------
3498:    lxi  h,$4b70           #       v4b70_wingformation-base
(34a6)
349B:    push h
349C:       call $35b0
349F:    pop  h
34A0:    mov  a,l
34A1:    adi  $08
34A3:    mov  l,a
34A4:    cpi  $90
34A6:    jnz  $349b
34A9:    ret
```

```
3427, 3458
------------------------------------------------------
34AA:    lxi  h,$4b90          #wingdata
(34ad)
34AD:    push h
34AE:       call $35b0
34B1:    pop  h
34B2:    mov  a,l
34B3:    adi  $08
34B5:    mov  l,a
34B6:    cpi  $b0
34B8:    jnz  $34ad         # until 4bb0
34BB:    ret


21f9, 3478, 348a
#Wing of bigbird beneath logo?
------------------------------------------------------
#   4b70 bytes: [W,X,Y,Z]  {W=index, XY=pointer bg-schermpos, Z=offset} vb [02,49,EF,01]
34C0:    mov  a,m           #   (4b70 =index)
34C1:    ana  a             #
34C2:    rz                 #   geen animatie todo meer bij W=0

34C3:    mov  b,a           #       copietje van index W
34C4:    adi  $c0           #
34C6:    mov  e,a           #
34C7:    mvi  d,$3e         #
34C9:    ldax d             #   [A] := 3ec0[W]  vb bij W=2 -> 40
34CA:    mov  c,a           #   vb BC=0240
34CB:    inr  l             #
34CC:    mov  d,m           #
34CD:    inr  l             #
34CE:    mov  e,m           #   [DE] bevat nu XY pointer, vb 49EF
34CF:    inr  l             #
34D0:    mov  a,b           #   index W
34D1:    rlc                #
34D2:    rlc                #
34D3:    rlc                #   index W*8    02*8 = 10
34D4:    add  m             #   4b73 (Z) bevat offset   vb [A:=11]
34D5:    ani  $7e           #   0111.1110
34D7:    mov  l,a           #
34D8:    mvi  h,$3e         #   VAR= HL=3e.[(W*8 + Z)and_7E]   (vb 3e10)
34DA:    mov  a,m           #
34DB:    inr  l             #
34DC:    mov  l,m           #
34DD:    mov  h,a           #   HL := * (VAR)

"   BIG BIRD Sprites, hatching out of an egg, flying wings and reverse back to an egg


    offset1   (W is index) bepaalt lowbyte voor [BC] return value voor op de stack
              voorbeeld: 4b70:[06 49 ef 00] geeft 06 voor W, 30c0+6 -> 30
              op de stack komt nu $3530, dit bepaalt de hoeveelheid bytetransfer
        W=  0   1  2  3  4  5  6  7  8 9   a  b  c  d  e  f
    3EC0:  FF  48 40 40 40 38 30 28 38 30 28 20 30 20 30 28  entry 35c0[+W]
          (#    4  6  6  6  8  a  c  8  a  c  e  a  e  a  c  numbytes transfer)
        ?
        3ED0:  01 01 01 01 00 00 01 01 00 01 01 01 00 00 00 01  ??[10 <= W <= 1F]
        3EE0:  05 04 03 02 01 00 00 00 00 00 01 01 01 01 02 02  ??[20 <= W <= 2F]
        3EF0:  02 02 03 03 03 04 04 04 05 05 06 06 07 08 07 06  ??[30 <= W <= 3F]

        ?
        3E00:  [01 02 04 08 10 20 40 80]

    HL lookup table  VAR(W*8 + Z and 7E is index) onderstaande zijn pointerpaartjes
    mbt uiteindelijke waarde voor bron(DE na xchg) van de byte copy
    3e08:   (W=1) 3DA8 3DAC 3DB0 3DB4
    3E10:   (W=2) 3D90 3D96 3D9C 3DA2  (W=3) 3D78 3D7E 3D84 3D8A
    3E20:   (W=4) 3D60 3D66 3D6C 3D72  (W=5) 3D40 3D48 3D50 3D58
    3E30:   (W=6) 3D18 3D22 3D2C 3D36  (W=7) 3CC0 3D00 3D0C 3C00
```

```
    3E40:   (W=8) 3D58 3D50 3D48 3D40  (W=9) 3D36 3D2C 3D22 3D18
    3E50:   (W=a) 3C00 3D0C 3D00 3CC0  (W=b) 3C00 3C0E 3C1C 3C2A
    3E60:   (W=c) 3C38 3C42 3C4C 3C56  (W=d) 3C60 3C6E 3C7C 3C8A
    3E70:   (W=e) 3C98 3CA2 3CAC 3CB6  (W=f) 3CC0 3CCC 3CD8 3CE4



#star egg
1:4 3da8: [00 00 01 00]         3dac: [00 00 08 00]
    3db0: [00 00 0A 00]         3db4: [00 00 0B 00]

    3da8 visual  3dac visual  3db0 visual  3db4 visual
    [00 01]      [00 08]      [00 0A]      [00 0B]
    [00 00]      [00 00]      [00 00]      [00 00]

#small egg
2:6 3d90: [00 00 90 00 00 00]   3d96: [00 00 91 00 00 00]
    3d9c: [00 00 92 00 93 00]   3da2: [00 00 94 00 95 00]

    [00 90 00]    [00 91 00]  [00 92 93 ]  [00 94 95]
    [00 00 00]    [00 00 00]  [00 00 00 ]  [00 00 00]


3:6 3D78: [00 00 96 00 00 00]   3D7E: [00 00 97 00 93 00]
    3D84: [00 00 98 00 99 00]   3D8A: [00 00 9A 00 9B 00]

4:6 3D60: [00 00 9C 00 00 00]   3D66: [00 00 9D 00 9E 00]
    3D6C: [00 00 9F 00 A0 00]   3D72: [00 00 A1 00 A2 00]



wing out
5:8 3D40: [00 00 9D 00 9E 00 00 00]  3D48: [00 00 9F 00 A0 00 00 00]
    3D50: [00 00 00 00 9C 00 00 00]  3D58: [00 00 00 00 A3 A5 A4 A6] egg-out


6:a 3D18: [00 00 00 00 A7 A9 A8 AA 00 00]   3D22: [00 00 00 00 AB AD AC AE 00 00]
    3D2C: [00 00 DE 00 AB B0 AC B1 DF 00]   3D36: [00 00 DE E0 AB B2 AC B3 DF E1]

more wing out
7:c 3CC0: [00 00 DE E2 AB B2 AC B3 DF E3 00 00]  3D00: [00 00 FA FC D7 D9 D8 DA FB FD 00 00]
    3D0C: [F4 F6 F5 00 C4 C6 C5 C7 F7 00 F8 F9]  3C00: [E8 00 E9 00 C4 C6 C5 C7 EA 00 EB 00]

wing in
8:8 3D58: [00 00 00 00 A3 A5 A4 A6]   3D50: [00 00 00 00 9C 00 00 00]
    3D48: [00 00 9F 00 A0 00 00 00]   3D40: [00 00 9D 00 9E 00 00 00]

wing out
9:a 3D36: [00 00 DE E0 AB B2 AC B3 DF E1]   3D2C: [00 00 DE 00 AB B0 AC B1 DF 00]
    3D22: [00 00 00 00 AB AD AC AE 00 00]   3D18: [00 00 00 00 A7 A9 A8 AA 00 00]

more wing out
a:c 3C00: [E8 00 E9 00 C4 C6 C5 C7 EA 00 EB 00]   3D0C: [00 00 EC 00 E9 00 C8 CA C9 CB EA 00]
    3D00: [00 00 FA FC D7 D9 D8 DA FB FD 00 00]   3CC0: [00 00 DE E2 AB B2 AC B3 DF E3 00 00]

    3c00 visual          d0c visual          3d00 visual         3cc0 visual
    [E8 E9 C4 C5 EA EB]  [00 EC E9 C8 C9 EA] [00 FA D7 D8 FB 00] [00 DE AB AC DF 00]
    [00 00 C6 C7 00 00]  [00 00 00 CA CB 00] [00 FC D9 DA FB 00] [00 E2 B2 B3 E3 00]


even more wing out
b:e 3C00: [E8 00 E9 00 C4 C6 C5 C7 EA 00 EB 00 00 00]
    3C0E: [EC 00 E9 00 C8 CA C9 CB EA 00 ED 00 00 00]
    3C1C: [EE 00 EF 00 CC CF CD D0 CE D1 F0 00 F1 00]
    3C2A: [F2 00 EF 00 D2 00 D3 D5 D4 D6 F0 00 F3 00]

wing in
c:a 3C38: [E8 00 E9 00 C4 C6 C5 C7 00 00]   3C42: [EC 00 E9 00 C8 CA C9 CB 00 00]
    3C4C: [EE 00 EF 00 CC CF CD D0 DD D1]   3C56: [F2 00 EF 00 D2 00 D3 D5 DD D6]

wing out
```

```
d:e 3C60: [00 00 00 00 C4 C6 C5 C7 EA 00 EB 00 00 00]
    3C6E: [00 00 00 00 DB CA C9 CB EA 00 ED 00 00 00]
    3C7C: [00 00 00 00 DC CF CD D0 CE D1 F0 00 F1 00]
    3C8A: [00 00 00 00 00 00 D3 D5 D4 D6 F0 00 F3 00]

wing in
e:a 3C98: [00 00 00 00 C4 C6 C5 C7 00 00]   3CA2: [00 00 00 00 DB CA C9 CB 00 00]
    3CAC: [00 00 00 00 DC CF CD D0 DD D1]   3CB6: [00 00 00 00 00 00 D3 D5 DD D6]

wing out
f:c 3CC0: [00 00 DE E2 AB B2 AC B3 DF E3 00 00]   3CCC: [00 00 00 E5 B4 B6 B5 B7 E4 E6 00 00]
    3CD8: [00 00 00 00 B8 BB B9 BC BA BD 00 00]   3CE4: [00 00 00 00 BE C1 BF C2 C0 C3 00 E7]

"

34de: 38b1
------------------------------------------------------
34DE:    mov  a,d
34DF:    cpi  $4b
34E1:    jnz  $350c

34E4:    mov  a,e
34E5:    cpi  $50
34E7:    jc   $350c            # < 50

34EA:    mvi  b,$08            # 08
34EC:    inr  l
34ED:    inr  l
34EE:    sui  $20              #   kolomlengte
34F0:    mov  e,a
34F1:    cpi  $50
34F3:    jc   $3509            # < 50

34F6:    mvi  b,$10            # 10
34F8:    inr  l
34F9:    inr  l
34FA:    sui  $20              #   kolomlengte
34FC:    mov  e,a
34FD:    cpi  $50              # < 50
34FF:    jc   $3509

3502:    mvi  b,$18            # 18
3504:    inr  l
3505:    inr  l
3506:    sui  $20              #   kolomlengte
3508:    mov  e,a

(34f3,34ff)
3509:     mov  a,c
350A:     add  b
350B:     mov  c,a

(34e1,34e7)
350C:    mvi  b,$35            # 35.[C] vb 3540
350E:    push b                # op de stack, ret gaat naar deze locatie!
350F:     lxi  b,$ffdf
3512:     xchg                 #   HL <-> DE
3513:     mvi  m,$00           # clear XY schermpos
3515:     inx  h
3516:     mvi  m,$00           # en volgende
3518:     dad  b               #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
3519:    ret                   # transfer 6 bytes van  DE naar HL


------------------------------------------------------NN bytes verplaatsen + clear2 + cursorR
<20:3519 0e bytes>

"  + nieuwe HL bepalen via dab d  (BC=FFDF is -21 voor cursor naar rechts en hoog)

  //           0   1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
  //    3EC0:  FF  48 40 40 40 38 30 28 38 30 28 20 30 20 30 28  0n
```

```
  //
  //val  20  28  30  38  40  48
  //    [1] [3] [5] [7] [ 9] [11 13]  [0]
  //    [2] [4] [6] [8] [10] [12 14]  [0]
"

3520:     ldax d            #                  2 bytes verplaatsen
3521:     mov  m,a      # *(HL) :=  *(DE)
3522:     inx  d            #
3523:     inx  h         #
3524:     ldax d         #
3525:     mov  m,a      # *(HL+1) :=  *(DE+1)
3526:     inx  d         #
3527:     dad  b            # HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

0be5, <28:3519 0c bytes>
-----------------------------------------------------kolom 2 byte plot
#  2 bytes verplaatsen van (DE) naar (HL)
3528:     ldax d
3529:     mov  m,a      # *(HL) :=  *(DE)
352A:     inx  d         #
352B:     inx  h         #
352C:     ldax d         #
352D:     mov  m,a      # *(HL+1) :=  *(DE+1)
352E:     inx  d         #
352F:     dad  b            # HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

<30:3519 0a bytes>
-----------------------------------------------------kolom 2 byte plot
#  2 bytes verplaatsen van (DE) naar (HL)
3530:     ldax d
3531:     mov  m,a      # *(HL+2) :=  *(DE+2)
3532:     inx  d         #
3533:     inx  h         #
3534:     ldax d         #
3535:     mov  m,a      # *(HL+3) :=  *(DE+3)
3536:     inx  d         #
3537:     dad  b            # HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

obdc, <38:3519 08 bytes>
-----------------------------------------------------kolom 2 byte plot
#  2 bytes verplaatsen van (DE) naar (HL)
3538:     ldax d
3539:     mov  m,a   #  *HL++ := *DE++
353A:     inx  d     #
353B:     inx  h     #
353C:     ldax d     #
353D:     mov  m,a   #
353E:     inx  d     #
353F:     dad  b        #       HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

(OFF4, <40:3519 06 bytes> )
 -----------------------------------------------------kolom 2 byte plot
#  2 bytes verplaatsen van (DE) naar (HL)
3540:     ldax d
3541:     mov  m,a
3542:     inx  d
3543:     inx  h
3544:     ldax d
3545:     mov  m,a
3546:     inx  d
3547:     dad  b    #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

0BEE, <48:3519 04 bytes>
  -----------------------------------------------------kolom 4 byte plot
#  4 bytes verplaatsen van (DE) naar (HL)
3548:    ldax d
3549:    mov  m,a
354A:    inx  d
354B:    inx  h
354C:    ldax d
```

```
354D:    mov  m,a
354E:    inx  d
354F:    dad  b     #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

3550:    ldax d
3551:    mov  m,a
3552:    inx  d
3553:    inx  h
3554:    ldax d
3555:    mov  m,a
3556:    inx  d
3557:    dad  b     #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)

# reset 16bits value to zero
3558:    mvi  m,$00     # 2 bytes (HL en HL+1) := 00
355A:    inx  h
355B:    mvi  m,$00
355D:    ret


3421, 3442, 3455
--------------------------------------------------------
3560:       call $30aa      # fetch cyclic value in range [0-F]
3563:    mov  b,a
3564:    rlc
3565:    rlc
3566:    mov  c,a
3567:    rlc
3568:    rlc
3569:    ora  b
356A:    stax $436f
356D:    ldax $43b8         #        v43b8_staging0-F_videobit1
3570:    cpi  $40
3572:    jc   $3577

3575:    mvi  a,$30
(3572)
3577:    ani  $30
3579:    rrc
357A:    mov  b,a
357B:    ldax $v43bb_remaining_bigbird_count            # v43bb_remaining_bigbird_count
357E:    dcr  a             #     --1
357F:    cpi  $04
3581:    jc   $3586

3584:    mvi  a,$03
(3581)
3586:    rlc
3587:    ora  b
3588:    mov  b,a
3589:    ldax $439a         # lookup hi-val v439ab_16bits_counter
358C:    rlc                #     0xxx.xxxx
358D:    rlc                #     00xx.xxxx
358E:    ani  $20           #     00x0.0000
3590:    ora  b
3591:    adi  $80
3593:    mov  l,a
3594:    mvi  h,$3e         # 3E80+offset-b
3596:    mov  a,m
3597:    stax $436e         # 436e var ?
359A:    inr  l
359B:    mov  a,m
359C:    add  c
359D:    ani  $f8
359F:    stax $436d         # 436d var ?
35A2:    ret



349C, 34AE
--------------------------------------------------------
```

```
35B0:    mov  a,m       # HL=4b88
35B1:    ana  a
35B2:    rz

35B3:    mov  b,a
35B4:    inr  l
35B5:    inr  l
35B6:    inr  l
35B7:    inr  l
35B8:    mov  a,m       # HL=4b8c  (wingdata.state  (or 8bits counter? )
35B9:    ana  a
35BA:    jz   $35be

35BD:    dcr  m         # v4b8c--
(35ba)
# build-up stackrange (8 bytes)
35BE:    xchg
35BF:    push d
35C0:    mov  a,b
35C1:    rlc
35C2:    rlc
35C3:    rlc
35C4:    mov  l,a       #  offset = (B-index * 8)
35C5:    mvi  h,$3f     # 3F00 + index
35C7:    mov  b,m
35C8:    inx  h
35C9:    mov  c,m
35CA:    push b
35CB:    inx  h
35CC:    mov  b,m
35CD:    inx  h
35CE:    mov  c,m
35CF:    push b
35D0:    inx  h
35D1:    mov  b,m
35D2:    inx  h
35D3:    mov  c,m
35D4:    push b
35D5:    inx  h
35D6:    mov  b,m36c0
35D7:    inx  h
35D8:    mov  c,m
35D9:    push b
35DA:    xchg
35DB:    ret        # JUMP to last PUSH B (@35d9) value !!

"          #0                    #1[stackdata + jump  proceed]
    3F00:  FFFF FFFF FFFF FFFF     [20FF 02FF - 36D2  36C0]         #1
    3F10:  [20FF 03FF -36D2 35E0]  [30FF 04FF - 36D2  35E0]     #2  #3
    3F20:  [10FF 05FF -36EA 35E0]  [10FF 06FF - 36EA  36C0]     #4  #5
    3F30:  [1060 071F -370A 36C0]  [F010 0B1A - 370A  36C0]     #6  #7
    3F40:  [40FF 04FF -36EA 36C0]  [10FF 08FF - 36EA  36C0]     #8  #9
    3F50:  [4010 0F17 -370A 36C0]  [10FF 0AFF - 36EA  35E0]    #10 #11
    3F60:  [FFFF FFFF -36CC 35E0]  [FFFF FFFF - 36CC  35E0]    #12 #13
    3F70:  [10FF 06FF -36EA 35E0]  [1010 0779 - 370A  35E0]    #14 #15

    PUSH- jumptable: 36cc,36d2,36ea,370a (different values from  [ .. ..  -  jump  ..      ]
    PUSH- proceed:   35e0, 36c0          (different values from  [ .. ..  -  ..    proceed ]


    following jump table NOT confirmed !!
    ?   3F80:  [0148 EE00 -10B0 1020] [0149 2C00 -10A0 00B0]    #16 #17
    ?   3F90:  [0149 6A00 -1090 00B8] [0149 A800 -1080 00C0]    #18 #19
    ?   3FA0:  [0149 E600 -1070 00C8] [014A 2400 -1060 00C8]    #20 #21
    ?   3FB0:  [014A 6200 -1050 00C8] [014A A000 -1040 00C8]    #22 #23
    ?   3FC0:  [014A CE00 -1038 00B0] [0148 CC00 -10B8 1020]    #24 #25
    ?   3FD0:  [014A CA00 -1038 00B8] [0148 C800 -10B8 1018]    #26 #27
    ?   3FE0:  [014A C600 -1038 00C0] [0148 C400 -10B8 1010]    #28 #29
    ?   3FF0:  [014A C200 -1038 00C8] [0148 C000 -10B8 1008]    #30 #31
    3F80  01 48 EE 00 10 B0 10 20 01 49 2C 00 10 A0 00 B0   .Hî..°. .I,..·.°
    3F90  01 49 6A 00 10 90 00 B8 01 49 A8 00 10 80 00 C0   .Ij....¸.I¨....À
```

```
    3FA0  01 49 E6 00 10 70 00 C8 01 4A 24 00 10 60 00 C8   .Iæ..p.È.J$..`.È
    3FB0  01 4A 62 00 10 50 00 C8 01 4A A0 00 10 40 00 C8   .Jb..P.È.J ..@.È
    3FC0  01 4A CE 00 10 38 00 B0 01 48 CC 00 10 B8 10 20   .JÎ..8.°.HÌ..¸.
    3FD0  01 4A CA 00 10 38 00 B8 01 48 C8 00 10 B8 10 18   .JÊ..8.¸.HÈ..¸..
    3FE0  01 4A C6 00 10 38 00 C0 01 48 C4 00 10 B8 10 10   .JÆ..8.À.HÄ..¸..
"

[35DB] proceed address for PUSH- jumptable: 36cc,36d2,36ea,370a
------------------------------------------------------
35E0:    inr  l
35E1:    inr  l
35E2:    mov  a,m
35E3:    cpi  $10
35E5:    jnc  $3628

35E8:    mov  b,a
35E9:    dcr  l
35EA:    add  m
35EB:    mov  m,a
35EC:    dcr  l
35ED:    dcr  l
35EE:    mov  a,b
35EF:    add  m
35F0:    mov  m,a
35F1:    cpi  $08
35F3:    jc   $366a

35F6:    ani  $07
35F8:    mov  m,a
35F9:    dcr  l
35FA:    mov  a,m
35FB:    sui  $20
35FD:    mov  m,a
35FE:    jnc  $3604

3601:    dcr  l
3602:    dcr  m
3603:    inr  l
(35fe)
3604:    inr  l
3605:    inr  l
3606:    inr  l
3607:    mov  c,m
3608:    inr  l
3609:    inr  l
360A:    mov  a,m
360B:    dcr  l
360C:    mvi  m,$10
360E:    sub  c
360F:    jz   $3672

3612:    dcr  a
3613:    rrc
3614:    rrc
3615:    rrc
3616:    ani  $1f
3618:    cmp  b
3619:    inr  a
361A:    mov  m,a
361B:    rc

361C:    ldax $436e
361F:    mov  m,a
3620:    cmp  b
3621:    rz

3622:    inr  b
3623:    mov  m,b
3624:    ret
```

```
(35e5)
------------------------------------------------------
3628:    ani  $0f          # maintain low-end nibble
362A:    jz   $3744

362D:    mov  b,a
362E:    dcr  l
362F:    mov  a,m
3630:    sub  b
3631:    mov  m,a
3632:    dcr  l
3633:    dcr  l
3634:    mov  a,m
3635:    sub  b
3636:    mov  m,a
3637:    jnc  $3695

363A:    ani  $07
363C:    mov  m,a
363D:    dcr  l
363E:    mov  a,m
363F:    adi  $20
3641:    mov  m,a
3642:    jnc  $3648

3645:    dcr  l
3646:    inr  m
3647:    inr  l
(3642)
3648:    inr  l
3649:    inr  l
364A:    inr  l
364B:    mov  a,m
364C:    inr  l
364D:    inr  l
364E:    sub  m
364F:    rrc
3650:    rrc
3651:    rrc
3652:    ani  $1f
3654:    cmp  b
3655:    inr  a
3656:    dcr  l
3657:    jc   $3663

365A:    ldax $436e
365D:    cmp  b
365E:    jz   $3663

3661:    mov  a,b
3662:    inr  a
(3657,
3663:    ori  $10
3665:    mov  m,a
3666:    ret


(35f3)
------------------------------------------------------
366A:    mov  a,b
366B:    ana  a
366C:    rnz

366D:    inr  l
366E:    inr  l
366F:    inr  l
3670:    inr  m
3671:    ret

(360f)
------------------------------------------------------
```

```
3672:    dcr  l
3673:    mov  b,m
3674:    inr  l
3675:    inr  l
3676:    ldax $43c2         #           v43c2_SpaceShip-horizPixelpos    (spec.obj0.val2)
3679:    ani  $f8
367B:    cmp  b
367C:    jnc  $3680

367F:    mov  b,a
(367c)
 3680:    ldax $436d
 3683:    mov  c,a
 3684:    adi  $08
 3686:    stax $436d            #  increase 436d with 8
3689:    mov  a,b
368A:    sub  c
368B:    mvi  m,$08
368D:    rc

368E:    cpi  $08
3690:    rc

3691:    mov  m,a
3692:    ret


(3637)
------------------------------------------------------
# HL = 4b73 wingdata ( [ ww xx yy ZZ])
3695:   inr  l              # (wingdata.objN.val0 [ww])
3696:    inr  l             # wingdata.objN.val1 [xx] sprite_offset_0x1400
3697:    mov  b,m
3698:    inr  l             # (wingdata.objN.val2 [Yy])
3699:    inr  l             # wingdata.objN.val3 [Zz]   determines columnvalue of screenpos
369A:    mov  a,m
369B:    cmp  b
369C:    rnz                # return if a <> b

369D:    dcr  l             # wingdata.objN.val2 [yy]   determines rowvalue of screenpos
369E:    mvi  m,$00         #                           := 00
36A0:    inr  l
36A1:    ldax $43c2         #   v43c2_SpaceShip-horizPixelpos   (spec.obj0.val2)
36A4:    ani  $f8
36A6:    cmp  b
36A7:    jc   $36ab

36AA:    mov  b,a
(36a7)
 36AB:    ldax $436d
 36AE:    adi  $08
 36B0:    stax $436d        # increased 436d with 8
36B3:    add  b
36B4:    mvi  m,$c8        # *HL := c8
36B6:    rc

36B7:    cpi  $c8
36B9:    rnc

36BA:    mov  m,a
36BB:    ret



[35DB] proceed address for PUSH- jumptable: 36cc,36d2,36ea,370a
------------------------------------------------------
36C0:    mov  a,m   # v4b8c
36C1:    rrc
36C2:    rc         # bit #1 was set, skip each two steps
```

```
36C3:    dcr  l
36C4:    mov  a,m   # v4b8b_counter_0-7
36C5:    inr  a
36C6:    ani  $07   #   modulo 8 (0-7)
36C8:    mov  m,a   #4b73 (:= 01  [ww xx YY zz] of first enemy bird )
36C9:    ret


PUSH- jumptable: 36cc,36d2,36ea,370a
-----------------------------------------------------
36CC:    pop  d
36CD:    pop  b
36CE:    pop  h
36CF:    ret


PUSH- jumptable: 36cc,36d2,36ea,370a
----------------------------------------------------- sound related
36D2:    pop  d
36D3:    pop  b
36D4:    pop  h
36D5:    mov  a,m
36D6:    ana  a
36D7:    rnz

36D8:    mov  m,b
36D9:    dcr  l
36DA:    dcr  l
36DB:    dcr  l
36DC:    dcr  l
36DD:    mov  m,d
36DE:    ldax $4368
36E1:    ori  $01
36E3:    stax $4368
36E6:    ret

[35DB] & PUSH- jumptable: 36cc,36d2,36ea,370a
----------------------------------------------------- sound related
36EA:    pop  d
36EB:    pop  b
36EC:    pop  h
36ED:    mov  a,m
36EE:    ana  a
36EF:    rnz            #HL <> 0

36F0:    inr  l
36F1:    inr  l
36F2:    mov  a,m
36F3:    ani  $0f       # maintain low-end nibble
36F5:    rnz            #HL+2  = [1-F]

36F6:    dcr  l
36F7:    dcr  l
36F8:    mov  m,b       #HL    ==0
36F9:    dcr  l
36FA:    dcr  l
36FB:    dcr  l
36FC:    dcr  l
36FD:    mov  m,d       #HL-4
36FE:    ldax $4368
3701:    ori  $02
3703:    stax $4368
3706:    ret


[35DB] & PUSH- jumptable: 36cc,36d2,36ea,370a
----------------------------------------------------- sound related
370A:    pop  d
370B:    pop  b
370C:    pop  h
```

```
370D:    mov  a,m
370E:    ana  a
370F:    rnz              # return if A <> 0

3710:    inr  l
3711:    inr  l
3712:    mov  a,m
3713:    ani  $0f         # maintain low-end nibble
3715:    rnz

3716:    dcr  l
3717:    dcr  l
3718:    mov  m,b
3719:    dcr  l
371A:    dcr  l
371B:    dcr  l
371C:    dcr  l
371D:    mov  m,d
371E:    ldax $4368
3721:    ori  $04
3723:    stax $4368
3726:    ldax $436f
3729:    ana  e
372A:    ani  $f0         # only high nibble
372C:    rnz

372D:    mov  a,e
372E:    ani  $0f         # maintain low-end nibble
3730:    mov  m,a
3731:    inr  l
3732:    inr  l
3733:    inr  l
3734:    inr  l
3735:    mov  m,c
3736:    ldax $4368
3739:    ori  $08
373B:    stax $4368
373E:    ret
```

(362a)

```
#wingdata handling   4b76=11, --4b75, 4b73=07  .. 4b
3744:    mvi  m,$11
3746:    dcr  l
3747:    dcr  m
3748:    dcr  l
3749:    dcr  l
374A:    mvi  m,$07
374C:    dcr  l
374D:    mov  a,m
374E:    adi  $20       #wingdata column pos
3750:    mov  m,a
3751:    rnc

3752:    dcr  l
3753:    inr  m
3754:    ret
```

0FCF

```
# dead-anim slot (#2/#3) handling, (see assigning at 0fc0)
3758:    mov  a,m       # 4378 / 437c
3759:    ana  a
375A:    rz             # slot free (when first byte is 00)?

375B:    dcr  m         # dead-anim slot  first byte  value --
375C:    jz   $37cc     # reached 00? -> clear screen partly?
```

```
375F:    mov  a,m
3760:    rrc
3761:    jnc  $37b0

3764:    mvi  a,$0f
3766:    sub  m
3767:    ani  $0e
3769:    rlc            # swap of  high and low nibble
376A:    rlc             #
376B:    rlc             #
376C:    rlc             #  [xxxx][yyyy] -> [yyyy][xxxx]
376D:    inr  l
376E:    inr  l
376F:    mov  d,m
3770:    inr  l
3771:    mov  e,m
3772:    push a
3773:    push d
3774:    lxi  b,$ffdf   #       (-$21)
3777:       call $3796  #
377A:    pop  d         #
377B:    pop  a         #
377C:    cma            #   complement
377D:    mov  l,a       #
377E:    mvi  h,$ff     #
3780:    inx  h         #
3781:    dad  d         #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
3782:    xchg           #
3783:    lxi  h,$bfa0   #
3786:    dad  d         #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
3787:    rnc            #

3788:    xchg           #
3789:    lxi  d,$17d6   #
378C:    mvi  m,$00     #
378E:    inx  h         #
378F:    mvi  m,$00     #
3791:    dad  b         #   HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
3792:    jmp  $3540     #           6 bytes copy DE->HL


3777
------------------------------------------------------
3796:    adi  $60       #
3798:    mov  l,a       #
3799:    mvi  h,$00     #   0060?
379B:    jnc  $379f     #   ?
379E:    inr  h         #   0160?
(379b)                  #
379F:    dad  d         #   HL += DE 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
37A0:    xchg           #
37A1:    lxi  h,$bcc0   #
37A4:    dad  d         #   HL += DE 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
37A5:    rc             #

37A6:    xchg           #
37A7:    lxi  d,$17d0   # bird bird explosion animation base
37AA:    jmp  $3540     #           6 bytes copy DE->HL


(3761)
------------------------------------------------------
37B0:    inr  l
37B1:    mov  a,m
37B2:    daa            #bin2BCD conversion
37B3:    mov  m,a
37B4:    inr  l
37B5:    mov  d,m
37B6:    inr  l
37B7:    mov  e,m
```

```
37B8:    dcr  l
37B9:    dcr  l
37BA:    nop
37BB:       call $0217  #       cursor met 1 kolom positie naar rechts
37BE:    mvi  a,$20     #       '0'
37C0:    stax d         #
37C1:       call $0210  #       cursor met 1 kolom positie naar links
37C4:    mvi  b,$02     #       2 karakters
37C6:    jmp  $00c4     #       plot


(375c)
------------------------------------------------------------CLEAR screen (partly?)
37CC:    inr  l            #
37CD:    inr  l            #
37CE:    inr  l            #
37CF:    mov  a,m          #
37D0:    ani  $1f          # kolomlengte
37D2:    adi  $20          #
37D4:    mov  l,a          #
37D5:    mvi  h,$43        #    $4320 verwijzing?
37D7:    lxi  b,$ffdf      #    (-$21)
37DA:    lxi  d,$001a      #    #rows
(37e2)                     #
37DD:    mov  m,d          #  plot 00
37DE:    inx  h            #
37DF:    mov  m,d          #  plot 00
37E0:    dad  b            #    HL += BC 16 bits (B=ffdf=-$21/cursor 1xrechts,1xhoog)
37E1:    dcr  e             # 1a .. 01
37E2:    jnz  $37dd
37E5:    ret



3403, 3409, 39c3
------------------------------------------------------
3800:    ldax $43c4        #   (special object slot #1)
3803:    ani  $08
3805:    rz

3806:    ldax $43e6        #   ship laser projectile top ahead
3809:    adi  $08
380B:    mov  d,a
380C:    ldax $4bd2        # smallbird8_newpos?
380F:    mov  e,a
3810:    ldax $43e7
3813:    ani  $e0
3815:    mov  b,a
3816:    ldax $43e7
3819:    sub  e
381A:    nop
381B:    ani  $1f
381D:    ora  b
381E:    mov  e,a
381F:    ldax d
3820:    sui  $90
3822:    rc

3823:    mov  b,a
3824:    ldax $43c6        #  (spec.obj1.val3)
3827:    ani  $07
3829:    adi  $00
382B:    mov  l,a
382C:    mvi  h,$3e        # 3exx
382E:    mov  c,m
382F:    mov  a,e
3830:    ani  $0e
3832:    rlc
3833:    rlc
3834:    mov  e,a
3835:    mvi  a,$a8
```

```
3837:    sub  e
3838:    mov  e,a
3839:    mvi  d,$4b         #       4bxx
383B:    mov  a,b
383C:    cpi  $50
383E:       cc   $3844
3841:    jmp  $391c

383e
------------------------------------------------------
3844:    adi  $60
3846:    mov  l,a
3847:    mvi  h,$3b         # 3b60+x
3849:    mov  a,m
384A:    ana  c
384B:    rz

384C:       call $38a1
384F:    xchg
3850:    mov  a,m
3851:    mvi  m,$00
3853:    inr  l
3854:    inr  l
3855:    inr  l
3856:    inr  l
3857:    mov  d,m
3858:    pop  h
3859:    lxi  h,$v43bb_remaining_bigbird_count      #   v43bb_remaining_bigbird_count
385C:    dcr  m             #       --1
385D:    cpi  $0b
385F:    jc   $3894

3862:    mov  e,a
3863:    mvi  a,$ff
3865:    stax $4369         # v4369_BarrierShield-Hit := FF (bullet collision detected)
3868:    lxi  h,$4378
386B:    lxi  b,$1010
386E:    mov  a,e
386F:    cpi  $0f
3871:    jz   $38fb

3874:    mov  a,d
3875:    rrc
3876:    ani  $7c
3878:    adi  $30
387A:    mov  c,a
387B:    mov  a,e
387C:    cpi  $0e
387E:    jz   $38fb

3881:    mov  a,c
3882:    rrc
3883:    mov  c,a
3884:    mov  a,e
3885:    cpi  $0c
3887:    jnc  $38fb

388A:    mov  a,c
388B:    rrc
388C:    mov  c,a
388D:    jmp  $38fb


(385f)
------------------------------------------------------
3894:    lxi  b,$0d05
3897:    mvi  a,$ff
3899:    stax $4364             # v4364_enemy-hit-detected
389C:    jmp  $38f8
```

384C, 38C4

```
38A1:    push d
38A2:    mvi  c,$20
38A4:    xchg
38A5:    inx  h
38A6:    mov  d,m
38A7:    inx  h
38A8:    mov  e,m
38A9:    ldax $1992          # 0F?
38AC:    adi  $e1
38AE:    mov  l,a
38AF:    mvi  h,$17          # 17e1+x
38B1:       call $34de
38B4:    pop  d
38B5:    ret
```

(391f)

```
#[A=2b]
38BC:    adi  $b0
38BE:    mov  l,a
38BF:    mvi  h,$3b          # 3bb0+offset-A
38C1:    mov  a,m
38C2:    ana  c
38C3:    rz

    " off   #0  #1 #2 #3 ...
    3BB0:  [03] E0 03 E0 0F 80 0F 00 3C 00 1E 3F 00 FC F0 00"

38C4:       call $38a1
38C7:    ldax d
38C8:    sui  $0b
38CA:    jc   $38e9          # [A] < 0b

38CD:    cpi  $03
38CF:    jnc  $38e9

38D2:    mov  b,a
38D3:    mov  h,d
38D4:    mov  a,e
38D5:    adi  $05
38D7:    mov  l,a
38D8:    ldax $43c6          # special obj #1, 3th byte
38DB:    cmp  m              #                          < (HL = DE+05) -> Cflag=1
38DC:    ral                 #       *2 (C->D0 ->D1->D2..D7->C)
38DD:    rlc                 #       *4 (D7->D0->D1..)
38DE:    rlc                 #       *8
38DF:    ani  $04            # bit #3
38E1:    ora  b
38E2:    adi  $b8
38E4:    mov  l,a
38E5:    mvi  h,$3d          # 3db8+offset
38E7:    mov  a,m
38E8:    stax d
(38ca,38cf)
38E9:    mvi  a,$ff
38EB:    stax $4366          # v4366_BossShip-hit := FF (detected)
38EE:    lxi  b,$0702
38F1:    jmp  $38f8


    "   off    #0 #1 #2 #3 #4 #5 #6 #7
        3DB8:  0C 0C 0E FF 0D 0E 0D FF"
```

(38c9,38f1)

```
# if available (dead-anim) slot #0 @4370 or slot #1 @4374 then fill with hit collision position
```

```
38F8:    lxi  h,$4370       #  hl=4370
(3871,387E,3887,388D)
38FB:    xra  a
38FC:    cmp  m
38FD:    jz   $3906         #       when hl == 0

3900:    inr  l
3901:    inr  l
3902:    inr  l
3903:    inr  l
3904:    cmp  m
3905:    rnz                #       when hl+4 <> 0, no available free slot
(38fd)
3906:    mov  m,b           #   slot.val1 := b
3907:    inr  l
3908:    mov  m,c           #   slot.val2 := c
3909:    inr  l
390A:    ldax $43e6         #   SpaceShip laser projectile (top) ahead
390D:    mov  m,a           #   slot.val3 := *(43e6)   hi-byte ship laser projectile
390E:    inr  l
390F:    ldax $43e7
3912:    mov  m,a           #   slot.val4 := *(43e7)   low-byte ''''  top ahead
3913:    ldax $43c4         #   (special object slot #1)
3916:    ani  $f7           #       1111.0111 (bit #3 cleared)
3918:    stax $43c4         #   43c4 ANDed   (special object #1, 1st byte)
391B:    ret

(3841)
------------------------------------------------------
391C:    mov  a,b
391D:    cpi  $20
391F:    jnc  $38bc         # [A] >= 20
3922:    ret


(3a95)
-----------------------------------------------------------handling soundB?
3923:    rz                 # return if (v436b_8bitscounter ==0)

3924:    dcr  m             #  count down v436b_8bitscounter
3925:    mvi  l,$8d         #   v438d_SoundControlB
3927:    mov  a,m
3928:    ani  $3f           #                                       0011.1111
392A:    ori  $80           #   v438d_SoundControlB set bit #7      10xx.xxxx
392C:    mov  m,a
392D:    ret



3431, 3448
------------------------------------------------------
3930:    ldax $4bd2         # smallbird8_newpos?
3933:    ani  $1e
3935:    adi  $c0
3937:    mov  l,a
3938:    mvi  h,$3d         # 3dc0-3dde
393A:    mov  e,m
393B:    inr  l
393C:    mov  l,m
393D:    mvi  h,$4b         #   $4bxx verwijzing?
393F:       call $3a00
3942:    ldax $439f         # v439f_wing.YY_lowboundary
3945:    add  d
3946:    mov  c,a
3947:    ldax $439e         # v439e_wing.YY_hiboundary
394A:    sub  d
394B:    mov  b,a
(3956)
394C:    push h
394D:       call $395c
3950:    pop  h
```

```
3951:    mov  a,l
3952:    adi  $08
3954:    mov  l,a
3955:    dcr  e
3956:    jnz  $394c
3959:    ret


394d
------------------------------------------------------
395C:    mov  a,m
395D:    cpi  $05
395F:    rc

3960:    mov  a,l
3961:    adi  $05
3963:    mov  l,a
3964:    mov  a,m
3965:    cmp  b
3966:    rc

3967:    cmp  c
3968:    rnc

3969:    sui  $04
396B:    mov  b,a
396C:    dcr  l
396D:    dcr  l
396E:    dcr  l
396F:    ldax $4bd2             # smallbird8_newpos?
3972:    add  m
3973:    ani  $1f
3975:    rlc
3976:    rlc
3977:    rlc
3978:    adi  $08
397A:    mov  c,a
397B:    jmp  $25b7



340c
------------------------------------------------------
3980:    ldax $4bd2             # smallbird8_newpos?
3983:    sui  $0c
3985:    rc

3986:    cpi  $10
3988:    rnc

3989:    lxi  h,$43c4           #   (special object slot #1)
398C:    lxi  d,$4bc0           # BIRDPOS of ?
398F:    mvi  b,$04
3991:       call $05e0          #   copy [B] bytes van rom[HL] naar var[DE]
3994:    mvi  l,$e6             #   43e6 -> SpaceShip laser projectile (top) ahead
3996:    mvi  b,$02             #
3998:       call $05e0          #   copy [B] bytes van rom[HL] naar var[DE]
399B:    mvi  l,$e2             #   43e2
399D:    lxi  d,$43e6           #  SpaceShip laser projectile (top) ahead
39A0:    mvi  b,$02             #
39A2:       call $05e0          #   copy [B] bytes van rom[HL] naar var[DE]
39A5:    mvi  l,$c4             #   43c4
39A7:    mvi  m,$08             #       := 08
39A9:    lxi  d,$439e           #  v439e_wing.YY_hiboundary
39AC:    ldax $439b             #   v439ab_16bits_counter_low
39AF:    rrc                    #
39B0:    jc   $39bf             #
                                #
39B3:    inr  e                 #
39B4:    mvi  l,$e7             #   43e7
39B6:    mov  a,m
```

```
39B7:    sui  $20
39B9:    mov  m,a
39BA:    dcr  l
39BB:    mov  a,m
39BC:    sbi  $00
39BE:    mov  m,a
(39B0)
39BF:    ldax d
39C0:    stax $43c6             #  (spec.obj1.val3)
(39d8)
39C3:       call $3800
39C6:    lxi  h,$43c4           # 43c4     (special object slot #1)
39C9:    mov  a,m
39CA:    ani  $08
39CC:    jz   $39f0

39CF:    lxi  h,$43e7
39D2:    inr  m
39D3:    mov  a,m
39D4:    ani  $1f
39D6:    cpi  $1d
39D8:    jc   $39c3

(39fb)
------------------------------------------------------
39DB:    lxi  h,$4bc0           # BIRDPOS of ?
39DE:    lxi  d,$43c4           #  (special object slot #1)
39E1:    mvi  b,$04
39E3:       call $05e0          #   copy [B] bytes van rom[HL] naar var[DE]
39E6:    mvi  e,$e6             #
39E8:    mvi  b,$02             #
39EA:    jmp  $05e0             #   copy [B] bytes van rom[HL] naar var[DE]


(39CC)
------------------------------------------------------
39F0:    mvi  l,$a6             #   v43a6_barrier_repeat_timer
39F2:    mov  a,m
39F3:    cpi  $c0
39F5:    jc   $0cc4             # < c0

39F8:    sui  $01
39FA:    mov  m,a               # v43a6_barrier_repeat_timer --1
39FB:    jmp  $39db



393F
------------------------------------------------------
3A00:    ldax $v43bb_remaining_bigbird_count               #   v43bb_remaining_bigbird_count
3A03:    sui  $0c
3A05:    cma                    #       inverse
3A06:    inr  a
3A07:    mov  d,a
3A08:    ldax $439b
3A0B:    rrc
3A0C:    rrc
3A0D:    rc
3A0E:    pop  h
3A0F:    ret

(273f)
------------------------------------------------------
3A10:    lxi  h,$43b8           #    v43b8_staging0-F_videobit1
3A13:    mov  a,m               #
3A14:    ana  a                 #
3A15:    jnz  $3b43             #

3A18:    mvi  l,$8d             #   v438d_SoundControlB
3A1A:    mvi  m,$cf             #
3A1C:    ret
```

```
3b52
------------------------------------------------------
3A1D:    lxi  h,$4369          # v4369_BarrierShield-Hit
3A20:    mov  a,m              #
3A21:    ana  a                #
3A22:    jz   $3a40            # jump if v4369_BarrierShield-Hit not detected

3A25:    cpi  $20              #
3A27:    jc   $3a2c            #

3A2A:    mvi  m,$20            #
(3a27)
3A2C:    dcr  m                #
3A2D:    mov  a,m              #
3A2E:    rlc                   #
3A2F:    rlc                   #
3A30:    nop                   #
3A31:    cma                   #
3A32:    ani  $0e              #
3A34:    mvi  l,$8d            #    v438d_SoundControlB
3A36:    mov  m,a              #
3A37:    mvi  l,$68            #    4368
3A39:    mvi  m,$00            #        := 00
3A3B:    mvi  l,$66            #    v4366_BossShip-hit
3A3D:    mvi  m,$00            #        := 00
3A3F:    ret                   #

(3a22)
------------------------------------------------------
3A40:    mvi  l,$64             # v4364_enemy-hit-detected
3A42:    mov  a,m               #
3A43:    ana  a                 #
3A44:    jz   $3a62             # v4364_enemy-hit-detected == 0  (no)
                                #
3A47:    cpi  $10               #
3A49:    jc   $3a4e             # v4364_enemy-hit-detected < 10
                                #
3A4C:    mvi  m,$10             # v4364_enemy-hit-detected := 10

(3a49)
3A4E:    dcr  m                 # --v4364_enemy-hit-detected
3A4F:    mov  a,m               #
3A50:    rrc                    # 0F >> 0000.0111(1)     00 >> 0000.0000
3A51:    nop                    #
3A52:    nop                    #
3A53:    cma                    # invert 1111.1000     1111.1111
3A54:    ani  $07               # 0                    7
3A56:    ori  $10               # 10                     17
3A58:    mvi  l,$8c             # v438c_SoundControlA
3A5A:    mov  m,a               #   := 10               := 17
3A5B:    mvi  l,$66             # v4366_BossShip-hit
3A5D:    mvi  m,$00             #   := 00
3A5F:    ret



(3A44)
------------------------------------------------------v4364_enemy-hit-detected == 0 (no)
3A62:    mvi  l,$66             #   v4366_BossShip-hit
3A64:    mov  a,m               #
3A65:    ana  a                 #
3A66:    rz                     #   v4366_BossShip-hit == 0 (no)
                                #
3A67:    cpi  $10               #
3A69:    jc   $3a78             #   0 < v4366_BossShip-hit < 10
                                #
3A6C:    mvi  m,$10             #   v4366_BossShip-hit := 10
3A6E:    ldax $43b8             #    v43b8_staging0-F_videobit1
3A71:    ani  $08               #   0000.1000
3A73:    jz   $3a78             #
```

```
                                #
3A76:    mvi  m,$05             #   v4366_BossShip-hit := 05
                                #
(3a69,3A73)                     #
3A78:    dcr  m                 #   --v4366_BossShip-hit    (:= 04) (:=0F..00)
3A79:    mvi  l,$8c             #   v438c_SoundControlA
3A7B:    mov  a,m               #
3A7C:    ani  $08               #   0000.1000
3A7E:    ori  $04               #   0000.0100
3A80:    mov  m,a               #   v438c_SoundControlA:=04 of :=0c
3A81:    ret

3b58
-------------------------------------------------------
3A82:    lxi  h,$439a           # v439a_16bits_counter_hi
3A85:    mov  a,m
3A86:    cpi  $03
3A88:    rc

3A89:    mvi  l,$8d             #   v438d_SoundControlB
3A8B:    mov  a,m
3A8C:    ani  $3f
3A8E:    mov  m,a
3A8F:    ret

(3b5b)
-------------------------------------------------------
3A90:    lxi  h,$436b           # v436b_8bitscounter
3A93:    mov  a,m
3A94:    ana  a                 # Z flag = 00 when counter reached zero
3A95:    jmp  $3923


(23de,23e3,23fb)
-------------------------------------------------------- sound based on movement?
# sound related, based on $28 value of xx-sprite-offset?
#  run-mode (inserted a coin, level 1 just before starting to play)
3A98:    lxi  h,$4b70           #   v4b70_wingformation-base [W X Y Z]
3A9B:    lxi  b,$0800           # b=08 c=10 num elements?
3A9E:    lxi  d,$03b0           # d=inc3, e=untl 4bb0
(3ab2)
3AA1:    mov  a,m
3AA2:    inr  l                 # objN.1 (X=sprite_offset_0x1400)
3AA3:    ana  b                 # filter on (08)->bit #4
3AA4:    jz   $3aae             # bit set

3AA7:    mov  a,m
3AA8:    cpi  $28               # 40 decimal
3AAA:    jc   $3aae

3AAD:    inr  c
(3aa4,
3AAE:    mov  a,l
3AAF:    add  d                 # skip [D=3] bytes -> to start of next object (objN+1.0)
3AB0:    mov  l,a
3AB1:    cmp  e                 # until 4bb0
3AB2:    jnz  $3aa1

3AB5:    mov  a,c
3AB6:    ana  a                 # (Z when a=0, otherwise NZ)
3AB7:    rz

3AB8:    cpi  $08
3ABA:    jc   $3abf

3ABD:    mvi  a,$08
(3aba)
3ABF:    adi  $25
3AC1:    mov  c,a
3AC2:    lxi  h,$438c           # v438c_SoundControlA
3AC5:    mov  a,m
```

```
3AC6:    ani  $c0
3AC8:    ora  c
3AC9:    mov  m,a
3ACA:    ret



(23e8,23ed)
------------------------------------------------------
3AD0:    lxi  h,$438e          # 438e
3AD3:    mov  a,m
3AD4:    ani  $01
3AD6:    rlc
3AD7:    rlc
3AD8:    ori  $20
3ADA:    mov  b,a
3ADB:    dcr  l
3ADC:    mov  a,m
3ADD:    ani  $c0
3ADF:    ora  b
3AE0:    mov  m,a
3AE1:    mvi  l,$96            #       4396
3AE3:    mov  a,m
3AE4:    inr  m
3AE5:    ana  a                # (Z when a=0, otherwise NZ)
3AE6:    jz   $3af8

3AE9:    ldax $4bd6            # 4bd6
3AEC:    adi  $e0
3AEE:    mov  e,a
3AEF:    mvi  d,$3d            #       3dxx
3AF1:    ldax d
3AF2:    cmp  m
3AF3:    rnc

3AF4:    mvi  m,$00
3AF6:    ret


(3AE6)
------------------------------------------------------soundcontrol vars
3AF8:    mvi  l,$8e            #       438e
3AFA:    inr  m
3AFB:    dcr  l                #       v438d_SoundControlB
3AFC:    mov  a,m
3AFD:    ori  $10
3AFF:    mov  m,a
3B00:    ret


23f8
------------------------------------------------------
3B02:    lxi  h,$439a          # v439a_16bits_counter_hi
3B05:    mov  a,m
3B06:    cpi  $02
3B08:    rnc

3B09:    inr  l
3B0A:    mov  a,m
3B0B:    mov  b,a
3B0C:    ani  $60
3B0E:    mvi  l,$8d            #       v438d_SoundControlB
3B10:    mvi  m,$0a            #                           := 0a
3B12:    rnz

3B13:    mov  a,b
3B14:    ani  $02
3B16:    adi  $1c
3B18:    mov  m,a
3B19:    ret
```

```
3b4f
------------------------------------------------------
3B1B:    lxi  h,$4362           # 4362 (v4362_barrier_sound_timer)
3B1E:    mov  a,m
3B1F:    ana  a                 # (Z when a=0, otherwise NZ)
3B20:    rz
# 4362 <> 0
3B21:    cpi  $40
3B23:    jc   $3b28             # 4362 < $40
# 4362 >= $40
3B26:    mvi  m,$40             # 4362 := 40
(3b23)
3B28:    dcr  m                 # --4362
3B29:    mov  a,m
3B2A:    ani  $06
3B2C:    rlc
3B2D:    nop
3B2E:    mvi  l,$8d             #   v438d_SoundControlB
3B30:    mov  m,a
3B31:    ret


3b4c
------------------------------------------------------
3B33:    lxi  h,$436a           # 436a
3B36:    mov  a,m
3B37:    ana  a                 # (Z when a=0, otherwise NZ)
3B38:    rz

3B39:    dcr  m                 # 436a--
3B3A:    ani  $08               #
3B3C:    ori  $07
3B3E:    mvi  l,$8d             #   v438d_SoundControlB
3B40:    mov  m,a
3B41:    ret


(3A15)
------------------------------------------------------
3B43:    lxi  h,$43a4           #   v43a4_JT1-indexer
3B46:    mov  a,m               #
3B47:    cpi  $03               #   JumpTable4 index?
3B49:       cz   $23d6          #       call on zero (->JT4)

3B4C:       call $3b33
3B4F:       call $3b1b
3B52:       call $3a1d
3B55:       call $27bd
3B58:       call $3a82
3B5B:    jmp  $3a90


    3B5E:                                           FF FF
    3B60:  1F 7C F0 01 C0 07 7F FC F0 07 C0 1F FF FC 03 F0
    3B70:  0F C0 3F FC 1F F0 07 FE 3F F8 0F FF FF FC 1F FF
    3B80:  FC 1F FC 1F F0 7F F0 7F C0 FF 01 C0 FF 01 00 FF
    3B90:  07 00 FF 07 FC 1F FC 1F F0 7F F0 7F C0 FF 01 C0
    3BA0:  FF 01 00 FF 07 FF 07 FC 1F F8 0F F0 C0 03 FF FF

    3BB0:  03 E0 03 E0 0F 80 0F 00 3C 00 1E 3F 00 FC F0 00
    3BC0:  7F FE 00 F0 03 E0 00 00 0F 80 00 00 3F 00 FE 30
    3BD0:  00 06 FF 00 F8 00 00 03 E0 00 E0 08 20 04 C0 01
    3BE0:  E0 03 F8 0F 07 E0 3F 03 FF FF FF 3F FC FF F8 FF
    3BF0:  FF 07 E0 1F F0 FF FC FF 07 1E FC 1F 1F 7F FF FF

    #(0BE2 background chars)?
    3C00:  E8 00 E9 00 C4 C6 C5 C7 EA 00 EB 00 00 00 EC 00
    3C10:  E9 00 C8 CA C9 CB EA 00 ED 00 00 00 EE 00 EF 00
    3C20:  CC CF CD D0 CE D1 F0 00 F1 00 F2 00 EF 00 D2 00
    3C30:  D3 D5 D4 D6 F0 00 F3 00 E8 00 E9 00 C4 C6 C5 C7
```

```
3C40:  00 00 EC 00 E9 00 C8 CA C9 CB 00 00 EE 00 EF 00
3C50:  CC CF CD D0 DD D1 F2 00 EF 00 D2 00 D3 D5 DD D6
3C60:  00 00 00 00 C4 C6 C5 C7 EA 00 EB 00 00 00 00 00
3C70:  00 00 DB CA C9 CB EA 00 ED 00 00 00 00 00 00 00
3C80:  DC CF CD D0 CE D1 F0 00 F1 00 00 00 00 00 00 00
3C90:  D3 D5 D4 D6 F0 00 F3 00 00 00 00 00 C4 C6 C5 C7
3CA0:  00 00 00 00 00 00 DB CA C9 CB 00 00 00 00 00 00
3CB0:  DC CF CD D0 DD D1 00 00 00 00 00 00 D3 D5 DD D6
3CC0:  00 00 DE E2 AB B2 AC B3 DF E3 00 00 00 00 00 E5
3CD0:  B4 B6 B5 B7 E4 E6 00 00 00 00 00 00 B8 BB B9 BC
3CE0:  BA BD 00 00 00 00 00 00 BE C1 BF C2 C0 C3 00 E7
3CF0:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

3D00:  00 00 FA FC D7 D9 D8 DA FB FD 00 00 F4 F6 F5 00
3D10:  C4 C6 C5 C7 F7 00 F8 F9 00 00 00 00 A7 A9 A8 AA
3D20:  00 00 00 00 00 00 AB AD AC AE 00 00 00 00 DE 00
3D30:  AB B0 AC B1 DF 00 00 00 DE E0 AB B2 AC B3 DF E1
3D40:  00 00 9D 00 9E 00 00 00 00 00 9F 00 A0 00 00 00
3D50:  00 00 00 00 9C 00 00 00 00 00 00 00 A3 A5 A4 A6
3D60:  00 00 9C 00 00 00 00 00 9D 00 9E 00 00 00 9F 00
3D70:  A0 00 00 00 A1 00 A2 00 00 00 96 00 00 00 00 00
3D80:  97 00 93 00 00 00 98 00 99 00 00 00 9A 00 9B 00
3D90:  00 00 90 00 00 00 00 00 91 00 00 00 00 00 92 00
3DA0:  93 00 00 00 94 00 95 00 00 00 01 00 00 00 08 00
3DB0:  00 00 0A 00 00 00 0B 00 0C 0C 0E FF 0D 0E 0D FF

# (addressed from 3938)
3DC0:  06 70 07 70 08 70 08 70 08 70 07 78 06 80 05 88
3DD0:  04 90 03 98 02 A0 01 A8 02 70 03 70 04 70 05 70

3DE0:  40 40 40 40 40 40 40 34 2C 26 20 1C 18 14 12 0F
3DF0:  0D 0B 09 08 07 06 05 04 03 02 02 02 02 02 02 02

#hatched bigbird entry (3e08..)
3E00:  01 02 04 08 10 20 40 80 3D A8 3D AC 3D B0 3D B4
3E10:  3D 90 3D 96 3D 9C 3D A2 3D 78 3D 7E 3D 84 3D 8A
3E20:  3D 60 3D 66 3D 6C 3D 72 3D 40 3D 48 3D 50 3D 58
3E30:  3D 18 3D 22 3D 2C 3D 36 3C C0 3D 00 3D 0C 3C 00
3E40:  3D 58 3D 50 3D 48 3D 40 3D 36 3D 2C 3D 22 3D 18
3E50:  3C 00 3D 0C 3D 00 3C C0 3C 00 3C 0E 3C 1C 3C 2A
3E60:  3C 38 3C 42 3C 4C 3C 56 3C 60 3C 6E 3C 7C 3C 8A
3E70:  3C 98 3C A2 3C AC 3C B6 3C C0 3C CC 3C D8 3C E4

3E80:  05 40 05 20 04 30 04 10 06 48 06 28 05 38 05 18   (<--3594)
3E90:  07 50 07 30 06 40 06 20 08 58 08 38 07 48 07 28
3EA0:  06 10 05 20 05 30 05 40 08 18 07 28 07 38 06 48
3EB0:  08 20 07 30 07 40 07 50 08 30 08 40 08 50 08 60

#hatched bigbird offset table data 3ec1 = 01, 3ec2=02
3EC0:  FF [48 40 40 40 38 30 28 38 30 28 20 30 20 30 28  (<--34c7)
3ED0:  01 01 01 01 00 00 01 01 00 01 01 01 00 00 00 01
3EE0:  05 04 03 02 01 00 00 00 00 00 01 01 01 01 02 02
3EF0:  02 02 03 03 03 04 04 04 05 05 06 06 07 08 07 06]

3F00:  FF FF FF FF FF FF FF FF [20] FF [02] FF [36 D2 36 C0
3F10:  20] FF [03] FF [36 D2 35 E0 30] FF [04] FF [36 D2 35 E0
3F20:  10] FF [05] FF [36 EA 35 E0 10] FF [06] FF [36 EA 36 C0
3F30:  10 60 07 1F 37 0A 36 C0 F0 10 0B 1A 37 0A 36 C0
3F40:  40] FF [04] FF [36 EA 36 C0 10] FF [08] FF [36 EA 36 C0
3F50:  40 10 0F 17 37 0A 36 C0 10] FF [0A] FF [36 EA 35 E0]
3F60:  FF FF FF FF [36 CC 35 E0] FF FF FF FF [36 CC 35 E0]
3F70:  [10] FF [06] FF [36 EA 35 E0 10 10 07 79 37 0A 35 E0
3F80:  01 48 EE 00 10 B0 10 20 01 49 2C 00 10 A0 00 B0
3F90:  01 49 6A 00 10 90 00 B8 01 49 A8 00 10 80 00 C0
3FA0:  01 49 E6 00 10 70 00 C8 01 4A 24 00 10 60 00 C8
3FB0:  01 4A 62 00 10 50 00 C8 01 4A A0 00 10 40 00 C8
3FC0:  01 4A CE 00 10 38 00 B0 01 48 CC 00 10 B8 10 20
3FD0:  01 4A CA 00 10 38 00 B8 01 48 C8 00 10 B8 10 18
3FE0:  01 4A C6 00 10 38 00 C0 01 48 C4 00 10 B8 10 10
3FF0:  01 4A C2 00 10 38 00 C8 01 48 C0 00 10 B8 10 08]
============================================================== ende der ubung
```

```
" ===============================================================================================================
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=================================================================================================================
```

Phoenix's Scoring Bug Analyzed and Fixed   By Don Hodges

The program reserves 4 slots of memory for birds that have been hit by the player's missile.
Each of these slots are 4 bytes long,
and they are all contained in a single area of memory at locations #4370 through #437F.
[The # indicates a hexadecimal number.]
Slots 1 and 2 are normally used for the regular death animation for most of the birds.
Slots 3 and 4 are used for the special death animation,
which has the score shown in the middle of two shells which open on both sides of it and move apart.
This is used for small birds that are flying upwards with wings outstretched,
and for the large birds in the third and fourth stages.

Slot Number	Type	Byte 1	Byte 2	Byte 3	Byte 4	
1	Regular	4370	4371	4372	4373	regular hit
2	Regular	4374	4375	4376	4377	
3	Special	4378	4379	437A	437B	special hit ((200)) <-- ((200)) -->
4	Special	437C	437D	437E	437F	

Slots 1 and 2 are normally used for the regular death animation for most of the birds.
Slots 3 and 4 are used for the special death animation
which has the score shown in the middle of two shells which open on both sides of it and move apart.
This is used for small birds that are flying upwards with wings outstretched,
and for the large birds in the third and fourth stages.

In each slot, the first byte is used as a counter which indicates the state of the animation of the dying bird.
When a bird is hit, this byte is normally set to #C, or is set to #10 for the special animation.
This counter is decreased as the animation of the dying bird goes through its stages,
and reaches 0 (zero) when it is complete.

When a bird is hit, the program first checks to see which type of bird was hit.
If it was a normal bird, the memory location that is checked first is set to #4370,
which is the first byte of the first slot. If it is one that uses the special animation,
the first memory checked is set to #4378, which is the first byte of the third slot.

Then it examines the first byte from either slot 1 or slot 3. If the byte is zero,
this means that the slot is open to use and the code then uses this slot to store 4 bytes of data
that are related to the animation of the dying bird.

If the byte is not zero, this means that the slot is being used and the next slot is then checked.
The program then checks either slot 2 or 4. If the first byte is zero,
then it is then used, but if not then the code uses the next slot, without doing any more checks.

However if 3 birds flying upward are hit in quick succession,
then the program, after checking the third slot and the fourth slot,
ends up using 4 bytes starting at #4380 to store the data, in a nonexistent fifth slot.

The memory locations #4381, #4382, and #4383 are normally used to hold the player's score!
[The first byte #4380, does not appear to be used for anything.]
These bytes get overwritten. #4381 always gets the value #20.
This corresponds to the player's hundred thousands and ten thousands digits.
The second byte #4382 gets a value of #41 or #42.
This corresponds to the player's thousands and hundreds digits.
The last byte can vary widely, and represents the player's tens and ones digits.
When this occurs, the player's score becomes #204,XYZ,
where X can be 1 or 2 and Y and Z can be various digits.
[The score is held in memory in Binary Coded Decimal].
; come here when a bird has been hit.

```
0EB8: 21 78 43    lxi h,$4378 ; HL := #4378 [slot for special animation]
0EBB: 7A          mov a,d     ; A := D
0EBC: FE 10       cpi $10     ; Are we to use the special animation?
```

```
      0EBE: CA C3 0E    jz $0ec3    ; Yes, skip next step
      0EC1: 2E 70       mvi l,$70   ; Else HL := #4370 [slot for regular animation]
      0EC3: 7E          mov a,m     ; Load timer from this slot
      0EC4: A7          ana a       ; Is this slot available?
      0EC5: CA D5 0E    jz $0ed5    ; Yes, skip ahead, we will use this slot

      ; else check next slot ...
      0EC8: 2C          inr l
      0EC9: 2C          inr l
      0ECA: 2C          inr l
      0ECB: 2C          inr l       ; Increase HL by 4. [now at 2nd or 4th slot]
      0ECC: 7E          mov a,m     ; Load timer from this slot
      0ECD: A7          ana a       ; Is this slot available?
      0ECE: CA D5 0E    jz $0ed5    ; yes, skip ahead, we will use this slot

      ; else use the next slot.  Bugged when birds are flying upwards.
      ; source of 204K bug. HL becomes #4380 which is start of score.
      0ED1: 2C          inr l
      0ED2: 2C          inr l
      0ED3: 2C          inr l
      0ED4: 2C          inr l       ; Increase HL by 4 [now at 3rd or 5th slot]

      ; there should have been a check here to see if HL==#4380 and change it if so.

      0ED5: 72          mov m,d     ; Store D into byte 1
      0ED6: 2C          inr l       ; Next byte
      0ED7: 73          mov m,e     ; Store E into byte 2. score becomes 20xxxx
      0ED8: 2C          inr l       ; Next byte
      0ED9: 70          mov m,b     ; Store B into byte 3. score becomes 2041xx or 2042xx
      0EDA: ...
```

Sound is composed of many analog component and NE555 IC in various
situation and one melody chip.

From MAME melody chip allow to play 2 songs :

 - Jeux Interdits (played when beginning first stage)
 - La lettre ‡ Elise (played when finishing fifth stage)

See 'Phoenix_sound_explanation.txt' and 'Phoenix_sound_sheet.pdf' for more
information about analog parts.

Phoenix (Amstar) FPGA - DAR - 2016

Educational use only
Do not redistribute synthetized file with roms
Do not redistribute roms whatever the form
Use at your own risk

Update 2016 April 18 : Note

make sure to use phoenix.zip roms
MAME Phoenix (Amstar)

TV mode only RBG 15kHz
Vertical screen cabinet or Cocktail mode available

DE-35 top_level

PS2 keyboard input (same control keys for both players)
wm8731 sound output
NO board SRAM used
Uses pll for 18MHz and 11MHz generation from 50MHz

The original arcade hardware PCB contains 8 memory regions

 cpu addressable space

 - program   rom 16Kx8, cpu only access 0x0000 0x3fff

 - foreground tile map bank 1 ram   1Kx8, vid.gen. & cpu 0x4000 0x433f
   cpu working ram   ram   ,   0x4340 0x43ff
 - background tile map bank 1 ram   1Kx8, vid.gen. & cpu 0x4800 0x4b3f
   cpu working ram inc.stack   ram   ,   0x4b40 0x4bff

 - foreground tile map bank 2 ram   1Kx8, vid.gen. & cpu 0x4000 0x433f
   cpu working ram   ram   ,   0x4340 0x43ff
 - background tile map bank 2 ram   1Kx8, vid.gen. & cpu 0x4800 0x4b3f
   cpu working ram   ram   ,   0x4b40 0x4bff

 non cpu addressable region

 - foreground graphics   rom 2Kx16,
 - background graphics   rom 2Kx16,
 - color palettes   rom 128x6,

The pixel clock is 5.5MHz, the cpu (8085A) clock is 5.5MHz.

The game is based on two RAM tile_id maps, there is no sprite. Cpu manages all
graphix on its own except background scrolling.

video generator counts 352 pixels horizontal :
   96 pixels (12 tiles) non visible,
  256 pixels (32 tiles) visible

video generator counts 256 lines vertical :
   48 lines (6 tiles) non visible,
  208 lines (26 tiles) visible

Video generator scans 32(horizontal)x32(vertical) RAM address, only 0-25
vertical address contains tile_id data. Vertical address from 26 to 31
are used by cpu as working ram including stack.

There are 2 banks at the same address for foreground and also 2 banks for
background. Bank 1 is used to store graphix map for player one, bank 2 is
used to store graphix data for player two. Cpu control bank switching with
register at 0x7800 (data bit 0).

Working ram is made of 0xC0 (192) bytes in each bank of each map
(foreground/background). So there is 4 x 192 bytes available.

Video generator can be inverted when player 2 and cocktail mode active.
Register at 0x5800 contains a background horizontal counter offset to produce
background scrolling.

Background horizontal blanking start about one tile after foreground horizontal
blanking. This allow to keep clean the top most foreground text line and for
cpu to update this shadowed line with new data (MAME missed that point. So
background graphix appears suddenly at top of the screen). There is also
about one line less at bottom of the screen.

Remaining graphix stuff is as usual : tile_ids from RAM map address 2 graphix ROMs,
These 2 bytes from graphix ROMs are serialized thru shift register or mux. These
2 bits selects one color among 4 possibilities (00 = black). Color priority is
given to foreground over background. The color palette ROMs uses the 2 selector bits
and 3 more bits from the tile_id number itself and one more bit from register

at 0x7800 (data bit 1). Each palette ROM gives one bit for each color red,
blue, green. So finally there is 2bits for red, 2bits for blue and 2 bits for green.

    Palette color selector
    0 bit 5 of video ram value (divides 256 chars in 8 color sections)
    1 bit 6 of video ram value (divides 256 chars in 8 color sections)
    2 bit 7 of video ram value (divides 256 chars in 8 color sections)
    3 bit 0 of pixelcolor (either from CHAR-A or CHAR-B, depends on Bit5)
    4 bit 1 of pixelcolor (either from CHAR-A or CHAR-B, depends on Bit5)
    5 0 = CHAR-A, 1 = CHAR-B
    6 palette flag (see video control reg. bit#0)
    7 always 0

charset A pal 0 color-index 08 - 0F
        A   1 color   18 - 1F
        B   0 color   00 - 07
        B   1 color   10 - 17

        charset/palette A/0 A/1   FOREGROUND CHARS
        00-1F   color #8 #18   () alphanumeric (A..Z), chars like: [](*
        20-3F   color #9 #19   0..9,Copyright,?, spaceship (top)part
        40-5F   color #a #1a   (bottom)spaceship, laserbullets, enemybombs
        60-7F   color #b #1b   ...
        80-9F   color #c #1c
        A0-BF   color #d #1d
        C0-DF   color #e #1e
        E0-FF   color #f #1f

        charset/palette B/0 B/1   BACKGROUND CHARS
        00-1F   color #0 #10
        20-3F   color #1 #11
        40-5F   color #2 #12
        60-7F   color #3 #13
        80-9F   color #4 #14
        A0-BF   color #5 #15
        C0-DF   color #6 #16
        E0-FF   color #7 #17

palette selector in relation with charset char
FG
  col = (code >> 5);   //shift right to get bits #754 for colors (0..7)
  col = col | 0x08 | (m_palette_bank << 4) //+ pixelcolorselector A = 08 + $(0|1)0
BG
  code = m_videoram_pg[m_videoram_pg_index][tile_index + 0x800];
  col = (code >> 5);
  col = col | 0x00 | (m_palette_bank << 4);
Example:
  palette=1, charset A: char 8A gives color 1c
  (1000.1010)->0000.0100 | 08-> 0000.1100 | 0001.0000 -> 0001.1100 = $1c

---
Colors (RGB color composition values: 05,93,BA), A(alpha channel = FF)
050505 = (---) black
0505BA = (--B) light-blue
0593BA = (-gB)

930505 = (r--) dark-red
9305BA = (r-B)
93BA05 = (rG-)

BA0593 = (R-b)
BABA05 = (RG-)

BABABA = white

```
   |R r - |G g - |B b - |
   |------+------+------|-----------------
 1 |    05|    05|    05| BLACK
 2 |    05|    05|BA    | blue
 3 |    05|  93  |    05|    (green medium)?
 4 |    05|  93  |  93  |    (blue-cyan)?
 5 |    05|  93  |BA    |    (blue medium)?
 6 |    05|BA    |    05| GREEN
 7 |    05|BA    |  93  | cyan
 8 |    05|BA    |BA    | dark cyan
 9 |  93  |    05|    05|    (dark red)?
10 |  93  |    05|  93  |    (purple bright)?
11 |  93  |    05|BA    | dark purple
12 |  93  |  93  |    05| dark yellow
13 |  93  |  93  |  93  | grey
14 |  93  |BA    |    05|    (olive green)?
15 |  93  |BA    |BA    |    (cyan bluish)?
16 |BA    |    05|    05| RED
17 |BA    |    05|  93  |    (purple bordeaux)?
18 |BA    |    05|BA    | purple
19 |BA    |  93  |    05| dark orange
20 |BA    |  93  |  93  |     (pink alike)?
21 |BA    |  93  |BA    |     (purple medium)?
22 |BA    |BA    |    05| yellow
23 |BA    |BA    |BA    | WHITE

               Java    Mame
Color BLACK    000000  050505   1
Color WHITE    dbdbdb  BABABA  23
Color RED      ff0000  BA0505  16
Color GREEN    00ff00  05BA05   6
Color BLUE     2424db  0505BA   2
Color CYAN     00ffdb  05BA93   7
Color YELLOW   ffff00  BABA05  22
Color PINK     ffb6db  ?
Color ORANGE   ffb649          (not used?)
Color LTPURPLE ff24b6  ?
Color DKORANGE ffb600  BA9305  19
Color DKPURPLE b624ff  9305BA  11
Color DKCYAN   00dbdb  05BABA   8
Color DKYELLOW dbdb00  BABA05  12 ??
Color BLUISH   9595ff  9393BA
Color PURPLE   ff00ff  BA05BA  18




palette table aRGB colors (mame-m_save_pen) (alpha channel value is FF fully opaque)
#     00     01     02     03             04     05     06     07      color-index charset B - pal 0
000:  050505 050505 050505 050505   010:  050505 050505 050505 050505
020:  0505BA 0593BA 93BA05 9305BA   030:  BABA05 0593BA 0593BA 0593BA
040:  930505 BA0593 9305BA 059393   050:  0505BA BA0593 BA0593 BA0593
060:  939393 BA9305 BA9305 BA0593   070:  93BABA 93BA05 93BA05 93BA05

#     08     09     0a     0b             0c     0d     0e     0f      color-index charset A - pal 0
080:  050505 050505 050505 050505   090:  050505 050505 050505 050505
0A0:  05BA93 BA0505 BA0505 BA05BA   0B0:  BA05BA BA05BA BA0505 059305
0C0:  0593BA BABA05 BABA05 BA9393   0D0:  BA9393 BA9393 BABABA 930593
0E0:  05BABA BABABA BABABA BABA05   0F0:  BABA05 BABA05 BA05BA BABABA

#     10     11     12     13             14     15     16     17      color-index charset B - pal 1
100:  050505 050505 050505 050505   110:  050505 050505 050505 050505
120:  0505BA 0505BA 93BA05 9305BA   130:  BA0593 BA0593 BA0593 BA0593
140:  930505 BA0593 9305BA 059393   150:  0593BA 0593BA 0593BA 0593BA
160:  939305 BA9305 BA9305 BA0593   170:  93BA05 93BA05 93BA05 93BA05

#     18     19     1a     1b             1c     1d     1e     1f      color-index charset A - pal 1
180:  050505 050505 050505 050505   190:  050505 050505 050505 050505
1A0:  05BA93 BA0505 BA0505 05BA05   1B0:  05BA05 05BA05 BA05BA 05BA05
```

```
1C0:  0593BA BABA05 BABA05 BABA05   1D0:  BABA05 BABA05 BABABA 930593
1E0:  BABA05 BABABA BABABA BA93BA   1F0:  BA93BA BA93BA BABA05 BABABA



---  @TODO

      1
000:  BLACK BLACK BLACK BLACK   010:  BLACK BLACK BLACK BLACK
020:  BLUE 0593BA 93BA05 9305BA   030:  BABA05 0593BA 0593BA 0593BA
040:  930505 BA0593 9305BA 059393   050:  BLUE BA0593 BA0593 BA0593
060:  939393 BA9305 BA9305 BA0593   070:  93BABA 93BA05 93BA05 93BA05

      2
080:  BLACK BLACK BLACK BLACK   090:  BLACK BLACK BLACK BLACK
0A0:  CYAN  RED RED PURPLE   0B0:  PURPLE PURPLE RED 059305
0C0:  0593BA BABA05 BABA05 BA9393   0D0:  BA9393 BA9393 WHITE 930593
0E0:  DKCYAN WHITE WHITE BABA05   0F0:  BABA05 BABA05 PURPLE WHITE


100:  BLACK BLACK BLACK BLACK   110:  BLACK BLACK BLACK BLACK
120:  BLUE BLUE 93BA05 9305BA   130:  BA0593 BA0593 BA0593 BA0593
140:  930505 BA0593 9305BA 059393   150:  0593BA 0593BA 0593BA 0593BA
160:  939305 BA9305 BA9305 BA0593   170:  93BA05 93BA05 93BA05 93BA05

180:  BLACK BLACK BLACK BLACK   190:  BLACK BLACK BLACK BLACK
1A0:  CYAN RED RED 05BA05   1B0:  05BA05 05BA05 PURPLE 05BA05
1C0:  0593BA BABA05 BABA05 BABA05   1D0:  BABA05 BABA05 WHITE 930593
1E0:  BABA05 WHITE WHITE BA93BA   1F0:  BA93BA BA93BA BABA05 WHITE


------
// pallete x charset x character = color
/* 4 colors per pixel * 8 groups of characters * 2 charsets * 2 pallettes */
(00    01       10       11        00  01            10            11       PIXELvalue)
charset A palette 0
 BLACK BLACK    CYAN     CYAN,     bg, -           , Letters     , asterisks
 BLACK YELLOW   RED      WHITE,    bg, Ship middle , Numbers/Ship, Ship edge
 BLACK YELLOW   RED      WHITE,    bg, Ship middle , Ship        , Ship edge/bullets
 BLACK PINK     PURPLE   YELLOW,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK PINK     PURPLE   YELLOW,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK PINK     PURPLE   YELLOW,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK WHITE    PURPLE   YELLOW,   bg, Explosions
 BLACK PURPLE   GREEN    WHITE,    bg, Barrier
charset A palette 1
 BLACK BLUE     CYAN     CYAN,     bg, -           , Letters     , asterisks
 BLACK YELLOW   RED      WHITE,    bg, Ship middle , Numbers/Ship, Ship edge
 BLACK YELLOW   RED      WHITE,    bg, Ship middle , Ship        , Ship edge/bullets
 BLACK YELLOW   GREEN    PURPLE,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK YELLOW   GREEN    PURPLE,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK YELLOW   GREEN    PURPLE,   bg, Bird eyes   , Bird middle , Bird Wings
 BLACK WHITE    RED      PURPLE,   bg, Explosions
 BLACK PURPLE   GREEN    WHITE,    bg, Barrier

charset B palette 0
 BLACK RED      BLUE     WHITE,    bg, Starfield
 BLACK PURPLE   BLUISH   DKORANGE, bg, Planets
 BLACK DKPURPLE GREEN    DKORANGE, bg, turrets     , u-body      , l-body
 BLACK BLUISH   DKPURPLE LTPURPLE, bg, boss-face   , body        , feet
 BLACK PURPLE   BLUISH   GREEN,    bg, Eagles: face, body        , shell
 BLACK PURPLE   BLUISH   GREEN,    bg, Eagles: face, body        , feet
 BLACK PURPLE   BLUISH   GREEN,    bg, Eagles: face, body        , feet
 BLACK PURPLE   BLUISH   GREEN,    bg, Eagles: face, body        , feet
charset B palette 1
 BLACK RED      BLUE     WHITE,    bg, Starfield
 BLACK PURPLE   BLUISH   DKORANGE, bg, Planets
 BLACK DKPURPLE GREEN    DKORANGE, bg, turrets     , upper body  , lower body
 BLACK BLUISH   DKPURPLE LTPURPLE, bg, boss-face   , body        , feet
 BLACK BLUISH   LTPURPLE GREEN,    bg, Eagles: face, body        , shell
 BLACK BLUISH   LTPURPLE GREEN,    bg, Eagles: face, body        , feet
 BLACK BLUISH   LTPURPLE GREEN,    bg, Eagles: face, body        , feet
```

```
 BLACK BLUISH   LTPURPLE GREEN,    bg, Eagles: face, body        , feet

=======================================================
---------------
VHDL File list
---------------

rtl_dar/phoenix_de2.vhd         Top level for de2 board
rtl_dar/phoenix.vhd             Main logic

rtl_dar/pll50_to_11_and_18.vhd  PLL 11MHz and 18 MHz from 50MHz altera mf
rtl_dar/phoenix_video.vhd       Video genertor H/V counter, blanking and syncs
rtl_dar/phoenix_music.vhd       Melody
rtl_dar/phoenix_effect3.vhd     Sound effect 3
rtl_dar/phoenix_effect2.vhd     Sound effect 2
rtl_dar/phoenix_effect1.vhd     Sound effect 1

rtl_dar/phoenix_prog.vhd        Program PROM
rtl_dar/prom_palette_ic41.vhd   Palette PROM rbg high bit
rtl_dar/prom_palette_ic40.vhd   Palette PROM rbg low  bit
rtl_dar/prom_ic24.vhd           Graphix PROM background low  bit
rtl_dar/prom_ic23.vhd           Graphix PROM background high bit
rtl_dar/prom_ic40.vhd           Graphix PROM foreground low  bit
rtl_dar/prom_ic39.vhd           Graphix PROM foreground high bit

rtl_dar/gen_ram.vhd             Generic RAM (Peter Wendrich + DAR Modification)

wm_8731_dac.vhd                 DE1/DE2 audio dac

io_ps2_keyboard.vhd             Copyright 2005-2008 by Peter Wendrich (pwsoft@syntiac.com)
kbd_joystick.vhd                Keyboard key to player/coin input

rtl_T80/T80s.vhd                T80 Copyright (c) 2001-2002 Daniel Wallner (jesus@opencores.org)
rtl_T80/T80_Reg.vhd
rtl_T80/T80_Pack.vhd
rtl_T80/T80_MCode.vhd
rtl_T80/T80_ALU.vhd
rtl_T80/T80.vhd

----------------------
Quartus project files
----------------------
de2/phoenix_de2.qsf             de2 settings (files,pins...)
de2/phoenix_de2.qpf             de2 project

------------------------------
Required ROMs (Not included)
------------------------------
You need the following 14 ROMs binary files from phoenix.zip
(MAME Phoenix - Amstar)

b1-ic39.3b   prom_ic39.vhd
b2-ic40.4b   prom_ic40.vhd

ic23.3d      prom_ic23.vhd
ic24.4d      prom_ic24.vhd

ic45         prom_ic45.vhd
ic46         prom_ic46.vhd
ic47         prom_ic47.vhd
ic48         prom_ic48.vhd
h5-ic49.5a   prom_ic49.vhd
h6-ic50.6a   prom_ic50.vhd
h7-ic51.7a   prom_ic51.vhd
h8-ic52.8a   prom_ic52.vhd

mmi6301.ic40 prom_palette_ic40.vhd
mmi6301.ic41 prom_palette_ic41.vhd

------
Tools
```

You need to build vhdl files from the binary file :
 - Unzip the roms file in the tools/phoenix_unzip directory
 - Double click (execute) the script tools/make_phoenix_proms.bat to get the following files

prom_ic39.vhd
prom_ic40.vhd

prom_ic23.vhd
prom_ic24.vhd

phoenix_prog.vhdl

prom_palette_ic40.vhd
prom_palette_ic41.vhd

DO NOT REDISTRIBUTE THESE FILES

VHDL files are needed to compile and include roms directly into the project

The script make_phoenix_proms.bat uses make_vhdl_prom executables delivered both in linux and windows version. The script itself is delivered only in windows version (.bat) but should be easily ported to linux.

Source code of make_vhdl_prom.c is also delivered.

Compiling for de2

You can build the project with ROM image embeded in the sof file. DO NOT REDISTRIBUTE THESE FILES.
3 steps

 - put the VHDL ROM files into the project directory
 - build phoenix_de2
 - program phoenix_de2.sof

Keyboard and swicth

Use directional left/right key to move, space to fire,
up key for shield, F1 to start 1 player, F2 to start 2 players, F3 for coins.

DE2-board switches :

   0 - 7 : dip switch
            0-1 : lives 3-4-5-6
            3-2 : bonus life 30K-40K-50K-60K
              4 : coin 1-2
            6-5 : unkonwn
              7 : upright-cocktail

   8 -10 : sound_select
            0XX : all mixed (normal)
            100 : sound1 only
            101 : sound2 only
            110 : sound3 only
            111 : melody only

DE2-board key(s) :

      0 : reset

HIGH SCORE gerelateerde adressen
0x4380 score PL1 High BCD tot 4380+3 low BCD waarde (000000)
0x4384 score PL2 High BCD tot 4384+3 low BCD waarde (000000)
0x4388 hiscore   High BCD tot 4388+3 BCD low waarde (000000)
        pokeb(0x41e1, (peekb(0x4389) / 16)+0x20);
        pokeb(0x41c1, (peekb(0x4389) & 0xf)+0x20);
        pokeb(0x41a1, (peekb(0x438a) / 16)+0x20);
        pokeb(0x4181, (peekb(0x438a) & 0xf)+0x20);

```
        pokeb(0x4161, (peekb(0x438b) / 16)+0x20);
        pokeb(0x4141, (peekb(0x438b) & 0xf)+0x20);

---------
andere adressen?
$438F =09 (/99?) #Credits, infinite bij 99
$43A6  =DD  Never have shields, 43A6=00 Shield Always Ready
$43C0  =84  Infinite Shields?
$4390  =09  Infinite Lives PL1
$4391  =09  Infinite Lives PL2

Instruction RST 0 RST 1 RST 2 RST 3 RST 4 RST 5 RST 6 RST 7
Restart Address 0000H 0008H 0010H 0018H 0020H 0028H 0030H 0038H

The 8085 has four additional interrupts and these interrupts
generate RST instructions internally and thus do not require any external hardware.
These instructions and their Restart addresses are: Interrupt TRAP RST 5.5 RST 6.5 RST 7.5
Restart Address 0024H 002CH 0034H 003CH

SP: LB HB (little endian)


Some software data information :

  0x4381-0x4383 : Player1 score in BCD
  0x4385-0x4387 : Player2 score in BCD
  0x4389-0x438b : High    score in BCD
  0x438c-0x438d : Copy of sound control 0x6000 and 0x6800
  0x438f        : Coin counter
  0x4390-0x4391 : Player 1 and Player 2 life counters
  0x439a-0x439b : Maybe frame counter
  0x43a4        : Something to do with game phase (not sure)
                     0 = attract screen
                     1 = ...
                     2 = ...
                     3 = playing
                     4 = player ship distroyed
                     5 = game over screen

  0x43a5        : Timer count down of game phase

  0x4b70-0x4baf : During stage 1 and 2 there are 16 groups of 4 bytes with
                  something to do with birds state (quiet or flying or dead)
                  and birds positions

                  During stage 3 and 4 there are 8 groups of 8 bytes with
                  something to do with eagles state.

  0x4bff        : cpu stack, grow downwards
=-=-=-==-=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
"


Symbol table
============

Value       Type    Name
-----       ----    ----
0000        Code    L0000
0000        Data    X0000
0002        Code    L0002
0008        Code    L0008
0010        Code    L0010
0012        Code    L0012
0013        Code    L0013
0018        Code    L0018
001A        Code    L001A
0020        Code    L0020
0020        Data    X0020
0028        Code    L0028
002D        Code    L002D
```

0030	Code	L0030
0038	Code	L0038
0041	Code	L0041
0046	Code	L0046
0050	Code	L0050
006B	Code	L006B
0070	Code	L0070
0078	Code	L0078
0080	Code	L0080
0088	Code	L0088
00BB	Code	L00BB
00C4	Code	L00C4
00D2	Code	L00D2
00D3	Code	L00D3
00E1	Code	L00E1
00E3	Code	L00E3
00E6	Code	L00E6
00F0	Code	L00F0
00F3	Code	L00F3
0140	Code	L0140
0173	Code	L0173
0196	Code	L0196
01C0	Code	L01C0
01C1	Code	L01C1
01C8	Code	L01C8
01D0	Code	L01D0
01E1	Code	L01E1
01E4	Code	L01E4
01ED	Code	L01ED
0200	Code	L0200
0206	Code	L0206
0210	Code	L0210
0217	Code	L0217
0220	Code	L0220
0258	Code	L0258
0260	Code	L0260
0270	Code	L0270
0277	Code	L0277
0288	Code	L0288
02A7	Code	L02A7
02CB	Code	L02CB
02D4	Code	L02D4
02E3	Code	L02E3
02F0	Code	L02F0
0314	Code	L0314
0320	Code	L0320
032E	Code	L032E
0331	Code	L0331
0350	Code	L0350
0367	Code	L0367
0377	Code	L0377
0380	Code	L0380
0389	Code	L0389
03A0	Code	L03A0
03A6	Code	L03A6
03B0	Code	L03B0
03CE	Code	L03CE
03E2	Code	L03E2
03EB	Code	L03EB
03F1	Code	L03F1
0400	Code	L0400
0404	Code	L0404
041E	Code	L041E
0460	Code	L0460
0466	Code	L0466
0484	Code	L0484
0492	Code	L0492
04A0	Code	L04A0
04DF	Code	L04DF
04E6	Code	L04E6
04F4	Code	L04F4

```
04FB        Code    L04FB
0506        Code    L0506
0526        Code    L0526
0532        Code    L0532
0547        Code    L0547
0580        Code    L0580
05D8        Code    L05D8
05D9        Code    L05D9
05E0        Code    L05E0
05EC        Code    L05EC
05FA        Code    L05FA
0603        Code    L0603
0610        Code    L0610
062A        Code    L062A
0650        Code    L0650
0667        Code    L0667
067A        Code    L067A
0699        Code    L0699
06B0        Code    L06B0
06E8        Code    L06E8
06F0        Code    L06F0
0700        Code    L0700
0706        Code    L0706
0718        Code    L0718
0720        Code    L0720
0740        Code    L0740
07DC        Code    L07DC
07F0        Code    L07F0
07FF        Code    L07FF
0834        Code    L0834
0848        Code    L0848
085A        Code    L085A
0876        Code    L0876
0886        Code    L0886
088B        Code    L088B
08A0        Code    L08A0
08C4        Code    L08C4
08EA        Code    L08EA
08EB        Code    L08EB
08FF        Code    L08FF
0900        Code    L0900
0917        Code    L0917
0926        Code    L0926
0930        Code    L0930
0964        Code    L0964
096E        Code    L096E
097A        Code    L097A
09A0        Code    L09A0
09A6        Code    L09A6
09BA        Code    L09BA
0A50        Code    L0A50
0A56        Code    L0A56
0A6C        Code    L0A6C
0A72        Code    L0A72
0A8A        Code    L0A8A
0AA0        Code    L0AA0
0AD6        Code    L0AD6
0AD8        Code    L0AD8
0B15        Code    L0B15
0B48        Code    L0B48
0B95        Code    L0B95
0BA0        Code    L0BA0
0BBA        Code    L0BBA
0BCA        Code    L0BCA
0C00        Code    L0C00
0C10        Data    X0C10
0C40        Code    L0C40
0C56        Code    L0C56
0C59        Code    L0C59
0C6B        Code    L0C6B
0C71        Code    L0C71
```

```
0C84        Code    L0C84
0CB4        Code    L0CB4
0CC4        Code    L0CC4
0CD8        Code    L0CD8
0CDE        Code    L0CDE
0CF4        Code    L0CF4
0D1C        Code    L0D1C
0D22        Code    L0D22
0D30        Code    L0D30
0D4F        Code    L0D4F
0D5E        Code    L0D5E
0D70        Code    L0D70
0D76        Code    L0D76
0D86        Code    L0D86
0DBB        Code    L0DBB
0DCC        Code    L0DCC
0DD2        Code    L0DD2
0DDE        Code    L0DDE
0DF0        Code    L0DF0
0E10        Code    L0E10
0E39        Code    L0E39
0E45        Code    L0E45
0E58        Code    L0E58
0E70        Code    L0E70
0E7E        Code    L0E7E
0E90        Code    L0E90
0EA4        Code    L0EA4
0EAD        Code    L0EAD
0EC3        Code    L0EC3
0ED5        Code    L0ED5
0F03        Data    X0F03
0F23        Code    L0F23
0F38        Code    L0F38
0F56        Code    L0F56
0F58        Code    L0F58
0F63        Code    L0F63
0F74        Code    L0F74
0F92        Code    L0F92
0FA6        Code    L0FA6
0FC0        Code    L0FC0
0FD8        Code    L0FD8
1021        Data    X1021
14E0        Code    L14E0
17E0        Code    L17E0
183D        Data    X183D
198C        Data    X198C
1DF0        Code    L1DF0
1EE0        Code    L1EE0
1EE6        Code    L1EE6
2020        Data    X2020
2025        Code    L2025
2030        Code    L2030
2040        Code    L2040
2070        Code    L2070
2085        Code    L2085
2091        Code    L2091
20B0        Code    L20B0
20B1        Code    L20B1
20B5        Code    L20B5
20BF        Code    L20BF
20CA        Data    X20CA
20E1        Code    L20E1
20E8        Code    L20E8
211C        Code    L211C
2130        Code    L2130
2146        Code    L2146
2150        Code    L2150
2160        Code    L2160
2170        Code    L2170
2180        Code    L2180
2190        Code    L2190
```

```
21A5        Code    L21A5
21BA        Code    L21BA
21DC        Code    L21DC
2204        Code    L2204
222A        Code    L222A
2230        Data    X2230
224C        Data    X224C
2260        Code    L2260
227A        Code    L227A
227B        Code    L227B
2292        Code    L2292
22A3        Code    L22A3
22E0        Code    L22E0
22E2        Code    L22E2
22F0        Code    L22F0
22FA        Code    L22FA
2303        Code    L2303
231B        Code    L231B
2322        Code    L2322
2351        Code    L2351
237B        Code    L237B
2398        Code    L2398
23AC        Code    L23AC
23C0        Code    L23C0
23D6        Code    L23D6
242C        Code    L242C
246A        Code    L246A
2476        Code    L2476
2495        Code    L2495
24A0        Code    L24A0
24C4        Code    L24C4
24E0        Code    L24E0
24F2        Code    L24F2
2520        Code    L2520
253D        Code    L253D
2552        Code    L2552
2560        Code    L2560
2588        Code    L2588
2596        Code    L2596
25B7        Code    L25B7
25CA        Code    L25CA
25CD        Code    L25CD
25E0        Code    L25E0
2600        Code    L2600
2624        Data    X2624
2639        Code    L2639
263A        Code    L263A
2650        Code    L2650
2668        Code    L2668
2676        Code    L2676
267C        Code    L267C
2685        Code    L2685
2693        Code    L2693
269D        Code    L269D
26A3        Code    L26A3
26AA        Code    L26AA
26AE        Code    L26AE
26D0        Code    L26D0
26D9        Code    L26D9
26E4        Code    L26E4
26E5        Code    L26E5
2700        Code    L2700
2717        Code    L2717
2739        Code    L2739
2748        Code    L2748
2768        Code    L2768
2778        Code    L2778
27A8        Code    L27A8
27BD        Code    L27BD
27D8        Code    L27D8
27E2        Code    L27E2
```

27E9	Code	L27E9
2A13	Data	X2A13
2B3A	Data	X2B3A
2C3D	Data	X2C3D
2E2C	Data	X2E2C
2E80	Data	X2E80
3000	Code	L3000
305C	Code	L305C
3064	Data	X3064
3074	Code	L3074
3089	Code	L3089
309F	Code	L309F
30AA	Code	L30AA
30DA	Code	L30DA
30E4	Code	L30E4
30ED	Code	L30ED
30F2	Code	L30F2
3112	Code	L3112
313D	Code	L313D
314B	Code	L314B
3179	Code	L3179
318A	Code	L318A
3192	Code	L3192
31D6	Code	L31D6
3210	Code	L3210
3240	Code	L3240
324E	Code	L324E
328F	Code	L328F
32A4	Code	L32A4
32AB	Code	L32AB
32B0	Code	L32B0
3323	Data	X3323
3430	Data	X3430
3438	Code	L3438
3452	Code	L3452
3462	Code	L3462
3474	Code	L3474
3477	Code	L3477
3486	Code	L3486
3489	Code	L3489
3498	Code	L3498
349B	Code	L349B
34AA	Code	L34AA
34AD	Code	L34AD
34C0	Code	L34C0
34DE	Code	L34DE
3509	Code	L3509
350C	Code	L350C
3528	Code	L3528
3538	Code	L3538
3540	Code	L3540
3548	Code	L3548
3560	Code	L3560
3577	Code	L3577
3586	Code	L3586
35B0	Code	L35B0
35BE	Code	L35BE
3604	Code	L3604
3628	Code	L3628
3634	Data	X3634
3648	Code	L3648
3663	Code	L3663
366A	Code	L366A
3672	Code	L3672
3680	Code	L3680
3695	Code	L3695
36AB	Code	L36AB
3744	Code	L3744
3758	Code	L3758
3796	Code	L3796
379F	Code	L379F

```
37B0        Code    L37B0
37CC        Code    L37CC
37DD        Code    L37DD
3800        Code    L3800

383C        Data    X383C

3844        Code    L3844
3894        Code    L3894
38A1        Code    L38A1
38BC        Code    L38BC
38E9        Code    L38E9
38F8        Code    L38F8
38FB        Code    L38FB
3906        Code    L3906
391C        Code    L391C
3923        Code    L3923
3930        Code    L3930
394C        Code    L394C
395C        Code    L395C
3980        Code    L3980
39BF        Code    L39BF
39C3        Code    L39C3
39DB        Code    L39DB
39F0        Code    L39F0
3A00        Code    L3A00
3A10        Code    L3A10
3A1D        Code    L3A1D
3A2C        Data    X3A2C
3A2C        Code    L3A2C
3A40        Code    L3A40
3A4E        Code    L3A4E
3A62        Code    L3A62
3A78        Code    L3A78
3A82        Code    L3A82
3A90        Code    L3A90
3A98        Code    L3A98
3AA1        Code    L3AA1
3AAE        Code    L3AAE
3ABF        Code    L3ABF
3AD0        Code    L3AD0
3AF8        Code    L3AF8
3B02        Code    L3B02
3B1B        Code    L3B1B
3B28        Code    L3B28
3B33        Code    L3B33
3B43        Code    L3B43
3B4A        Data    X3B4A
3E3C        Data    X3E3C

4000        Code    L4000
4005        Code    L4005
4062        Data    X4062
4131        Code    L4131
4142        Data    X4142
42A2        Data    X42A2
4300        Data    X4300
431D        Data    X431D

4350        Data    X4350
4351        Data    X4351
4352        Data    X4352
4353        Data    X4353
4354        Data    X4354
4356        Data    X4356
4357        Data    X4357

4360        Data    X4360
4361        Data    X4361
4363        Data    X4363
4364        Data    X4364
```

4366	Data	X4366
4368	Data	X4368
4369	Data	X4369
436A	Data	X436A
436D	Data	X436D
436E	Data	X436E
436F	Data	X436F
438F	Data	X438F
4394	Data	X4394
4395	Data	X4395
4396	Data	X4396
4397	Data	X4397
439A	Data	X439A
439B	Data	X439B
439E	Data	X439E
439F	Data	X439F
43A0	Data	X43A0
43A2	Data	X43A2
43A3	Data	X43A3
43A4	Data	X43A4
43A5	Data	X43A5
43AA	Data	X43AA
43B3	Data	X43B3
43B8	Data	X43B8
43B9	Data	X43B9
43BA	Data	X43BA
43BB	Data	X43BB
43C2	Data	X43C2
43C3	Data	X43C3
43C4	Data	X43C4
43C6	Data	X43C6
43E6	Data	X43E6
43E7	Data	X43E7
4743	Code	L4743
488A	Data	X488A
4B50	Data	X4B50
4BD1	Data	X4BD1
4BD2	Data	X4BD2
4BD5	Data	X4BD5
4BD6	Data	X4BD6
4BD7	Data	X4BD7
5000	Data	X5000
5300	Code	L5300
5800	Data	X5800
6000	Data	X6000
600B	Code	L600B
642F	Code	L642F
6800	Data	X6800
7000	Data	X7000
7800	Data	X7800
B4B4	Code	LB4B4
BEC4	Code	LBEC4
C000	Code	LC000
C15E	Code	LC15E
C23F	Code	LC23F
C2E1	Code	LC2E1
C322	Code	LC322
C3C0	Code	LC3C0
C3C3	Code	LC3C3
C5C6	Code	LC5C6
C5D4	Code	LC5D4

```
CBC9        Code    LCBC9
CBDA        Code    LCBDA
CCFF        Code    LCCFF
CDCF        Code    LCDCF
CFD3        Code    LCFD3
D0D3        Code    LD0D3
D1DA        Code    LD1DA
D200        Code    LD200
D2CE        Code    LD2CE
D2DE        Code    LD2DE
D300        Code    LD300
D3E0        Code    LD3E0
D6C6        Code    LD6C6
D7FC        Code    LD7FC
D83C        Code    LD83C
DCFF        Code    LDCFF
DDCD        Code    LDDCD
E0E2        Code    LE0E2
E200        Code    LE200
E7E5        Code    LE7E5
E8F3        Code    LE8F3
E900        Code    LE900
EB00        Code    LEB00
EB4C        Code    LEB4C
ED00        Code    LED00
EF00        Code    LEF00
F003        Code    LF003
F01F        Code    LF01F
F0D6        Code    LF0D6
F1E6        Code    LF1E6
F5F6        Code    LF5F6
F81F        Code    LF81F
FBEB        Code    LFBEB
FC00        Code    LFC00
FC04        Code    LFC04
FC1F        Code    LFC1F
FCFC        Code    LFCFC
FDFB        Code    LFDFB
FDFC        Code    LFDFC
FF1F        Code    LFF1F
FFFF        Code    LFFFF
FFFF        Data    XFFFF

Number of symbols:          539
```