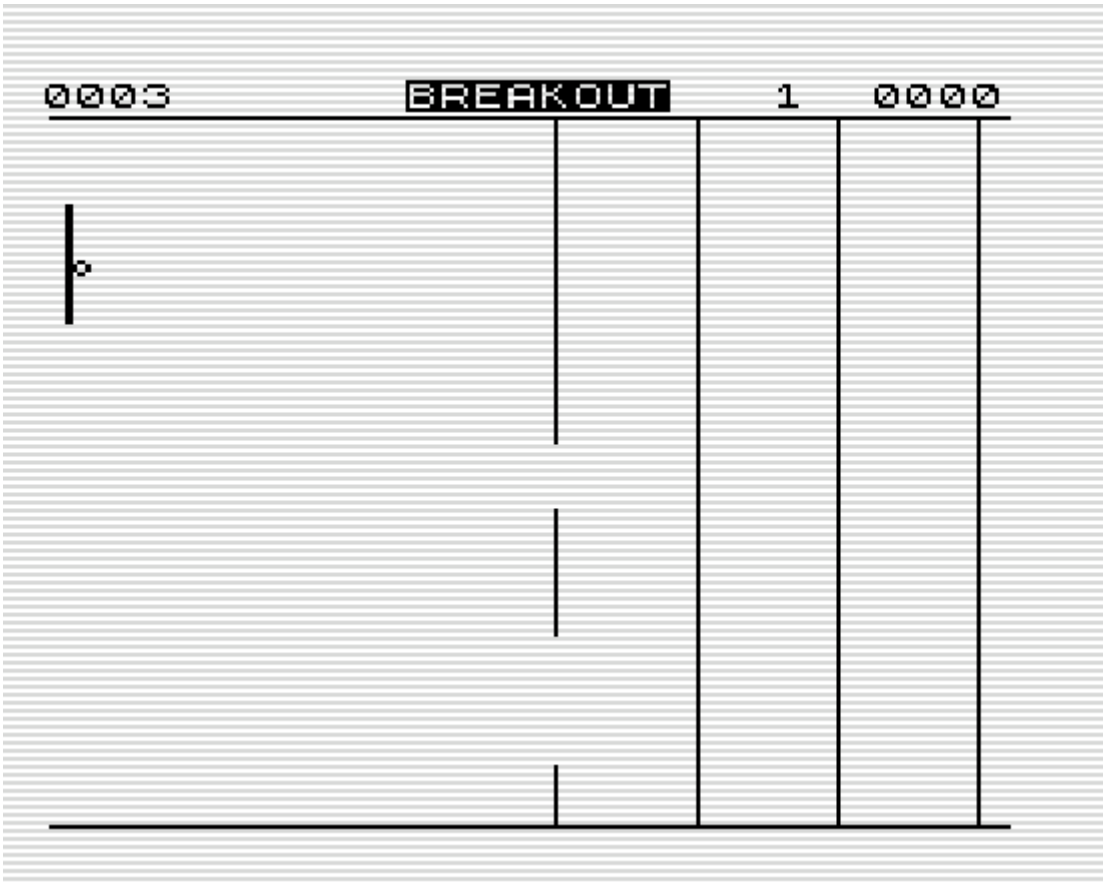
Break Out



The display of Break Out is a continuation of K-Bird but now the ball is moving, not the walls. The gameplay is a REAL Break Out. The sooner you reach the left side of the screen the more points you will score. There is no byte left to code. The stack is at minimum.
The hires routine has so much possibilities that it was almost undoable to fit all in 207 Tstates.

```
; Breakout
; A game with a TRUE breakout-theme

? * TORNADO *

            ORG  #4009              ;#4009
            DUMP 49161

b0          EQU  br0*256/256
b1          EQU  br1*256/256
b2          EQU  br2*256/256
b3          EQU  br3*256/256
b4          EQU  br4*256/256
b5          EQU  br5*256/256
b6          EQU  br6*256/256

; the single BASIC-line is fully coded
; over existing systemvariables
; linenumber and length is used as code

basic       LD   A,0                ; Line nr and
L400B       JR   init0              ; start of program

            DEFB 236                ; the BASIC-command
            DEFB 212,28,126         ; set in reusable sysvar
            DEFB 143,0,18           ; #4009 in FP notation
```

```
;d_file     DEFW 0
;dfcc       DEFW 0
;var        DEFW 0
;dest       DEFW 0

eline      DEFW last              ; needed on start
chadd      DEFW last-1
xptr       DEFW 0
stkbot     DEFW last
           DEFW last              ; memory above reused for data

berg       DEFB 0
mem        DEFW 0

init0      EX   AF,AF'
           JP   init

lastk      DEFB 255,255,255       ; used by ZX81
margin     DEFB 55                ; used by ZX81

nxtlin     DEFW basic
           DEFB 0
           DEFB 0

flagx      DEFB 0                 ; used by ZX81
strlen     DEFW 0

taddr      DEFW 3213              ; used by ZX81

seed       DEFW 0
frames     DEFW 65535             ; used by ZX81
coords     DEFB 0,0
prcc       DEFB 188
sposn      DEFB 33,24
cdflag     DEFB 64                ; fixed value

inudg      LD   A,(HL)            ; sync with outudg
           DEC  C
           LD   A,(HL)
           EXX
           JP   outudg+#8000

hr         LD   HL,dfile+#8000    ; we start with lowres
           LD   BC,#201
           LD   A,#1E
           LD   I,A
           LD   A,#FC
           CALL #2B5              ; display score

           LD   (savsp+1),SP      ; we have a screenstack

           LD   B,8
hr0        DJNZ hr0
           LD   A,(HL)

           LD   HL,lowret         ; faster RET from upmen
           EXX
           LD   HL,batdata        ; pointer to batpositions
           LD   BC,#B001          ; 176 lines, 1 to fetch bricks
           DEFB #DD
           LD   H,#10             ; 16 lines, set in highmem
           LD   A,#43             ; screen range
           LD   I,A
```

```
            LD   A,#5D
            LD   R,A
            CALL edge+#8000        ; show topline

            LD   SP,scrstack       ; bricks taken from stack

bally       LD   A,255             ; fetch ball y
            CP   B                 ; draw ballscreen?
            JR   Z,doball          ; if so do ballscreen

ballb2      LD   A,(HL)            ; is bat now on screen
            INC  HL                ; point to next bat
            SUB  B                 ; test with linenumber
            JP   Z,batbuff+#8000   ; if so, show bat
            DEC  HL                ; undo next pointer
            NOP                    ; timing
            LD   A,C               ; 1 brickline less
            SUB  1                 ; for timing like this
            JP   wallbuff+#8000    ; do wall display

batbuff     ADD  A,b0              ; LD doesn't work in upmem
            LD   R,A
            LD   A,C               ; done now or ballscreen
            ADD  A,255             ; couldn't sync
            DEFB 0                 ; show bat

wallbuff    JP   NZ,inudg          ; still in current brick
            DEFB #DD               ; reset bricklinecounter
            LD   C,H
            EXX
            POP  BC                ; fetch new UDG pointers
            POP  DE
; Test new brick is done in displaymemory, so all
; opcodes need bit 6 set to be executed

outudg      LD   A,C               ; get data brick 1
            LD   R,A
            NOP                    ; display brick 1
            LD   A,B               ; get data brick 2
            LD   R,A
            NOP                    ; display
            LD   A,E               ; brick 3
            LD   R,A
            NOP
            LD   A,D               ; brick 4
            LD   R,A
            NOP
            JP   (HL)              ; out of high memory (lowret)

lowret      EXX
            DJNZ bally             ; end of screen?

            POP  HL                ; timing RET above
            PUSH HL                ; POP/PUSH to save stack

            OR   (HL)              ; reset C in 7 tstates

            RET  C                 ; NEVER TRUE HERE

savsp       LD   SP,0              ; get original SP back

            LD   A,#5D
            LD   R,A
```

```
            CALL edge+#8000         ; show bottomline

            CALL #292               ; and back to program
            CALL #220
            LD   IX,hr
            JP   #2A4

; okudg  : 26 = 10 +  4 + 12 tstates
; doball : 23 =  7 +  4 + 12 tstates
okudg       NOP                     ; sync timing
            DEFB #CA                ; JP Z, never true
doball      LD   A,balldata*256/256
            NOP                     ; sync display with batbuf
            JR   rettime

ballback    OR   A                  ; skip ball test, reset C
            DJNZ ballb2             ; go to bat test
            JP   savsp-1            ; ball at bottom

balllow     RET  C                  ; here never true, timing
            DEC  B                  ; here NEVER 0
            DEC  C                  ; next brick??
            JR   NZ,okudg
            LD   C,16               ; in ballscreen
            EXX                     ; also get next
            POP  BC                 ; brickline data
            POP  DE
            EXX
rettime     JP   highball+#8000

dead        LD   HL,lives
            DEC  (HL)               ; decrease nr of lives
            LD   A,(HL)
            CP   28                 ; test against "0"
            JR   NZ,restart

eog         LD   HL,sc-1            ; check if new hiscore
            LD   DE,hi-1
            LD   BC,5
fihi        INC  HL
            INC  DE
            DEC  C
            JR   Z,start
            LD   A,(DE)
            CP   (HL)
            JR   Z,fihi
            JR   NC,start
            LDIR

start       LD   A,b0               ; reset startlevel
            LD   (mkscr+1),A

s1          LD   A,(lastk)
            SUB  %10111111          ; start with NL
            JR   NZ,s1

            LD   HL,#1C1C           ; reset score
            LD   (sc),HL
            LD   (sc+2),HL

            LD   A,31               ; "3" lives
            LD   (lives),A

nextlev     LD   HL,mkscr+1
```

```
            INC  (HL)
            LD   A,b6+1            ; All levels completed?
            CP   (HL)
            JR   Z,eog             ; if so end of game

            LD   HL,scrstack       ; make brick screen
            LD   B,44
mkscr       LD   (HL),b0
            INC  HL
            DJNZ mkscr

            LD   A,100
            LD   (baty+1),A

restart     LD   A,(baty+1)
            SUB  14
            LD   (bally+1),A

            LD   A,12              ; start ball x
            LD   (ballx+1),A

            XOR  A
            LD   (dxy+1),A         ; reset dx, A always 0

; after dead correct clearance needed
            LD   A,balldata*256/256
            LD   (erball+1),A

; for correct collision we need to erase the ball
; but also erase the bat or remainings can be shown
erball      LD   HL,balldata+2     ; position of ball
            LD   E,#89             ; startpos of bat
            LD   C,B               ; C is now SPACE
            LD   B,30              ; set batlength
er1         LD   (HL),C            ; erase part 1 of ball
            LD   A,L               ; save ballpointer
            INC  L
            LD   (HL),C            ; erase part 2 of ball
            LD   L,E               ; get batpointer
            LD   (HL),C            ; erase bat
            ADD  A,B               ; point to next ball
            LD   L,A               ; save ballposition
            LD   A,E               ; get bat
            ADD  A,B               ; point to next bat
            LD   E,A               ; save batposition
            JR   NC,er1            ; do full screen

; built the ballscreen, first set bricks and walls
            LD   A,(bally+1)       ; Bat can be on ballscreen
            LD   HL,batdata
baty        LD   E,0               ; fetch current batpos
setbat      LD   D,A               ; save current Y of ball
            SUB  E
            CP   5
            LD   C,A               ; C holds linenr on ballscreen
            JR   NC,notonball      ; set batdisplay
            PUSH HL
            LD   HL,balldata-29    ; we set on ballscreen
batball     INC  C
            LD   A,L
fbpos       ADD  A,30
            DEC  C
            JR   NZ,fbpos          ; calculate next ballline
            JR   C,balldone        ; ready when "out of screen"
```

```
            LD   L,A
            LD   (HL),%00110000    ; set bat
            DEC  E                 ; decrease batpointer
            DEC  B                 ; decrease batlength
            JR   Z,batdone         ; end of bat reached
            JR   batball           ; still on ball screen

balldone    POP  HL                ; end of ball reached
notonball   LD   A,D               ; get original Y back
            LD   (HL),E            ; set bat y in own index
            DEC  E
            INC  HL
nxtbat      DJNZ setbat            ; do length of bat

            DEFB 62                ; hide POP, already done
batdone     POP  HL
            LD   (HL),B            ; set impossible endmarker
            LD   A,(bally+1)

; bat is on ballscreen, now add bricks
            LD   DE,balldata+16    ; first brick position
            LD   B,A
brlp        LD   C,1               ; point to first column
            CALL brickfield        ; calculate brickpointer
            CALL getdata           ; fetch data of pointer
            LD   A,C
            ADC  A,A               ; 2x shift left
            ADC  A,A               ; is always possible
            LD   (DE),A            ; write wall to screen
            CALL deadd4            ; next column 4 further
            LD   A,C               ; also data of pointer
            LD   (DE),A            ; no shift, set data

            CALL deadd5            ; fetch pointer3
            CALL shift2            ; 2x shift right
            LD   (DE),A            ; also set over
            DEC  DE
            LD   A,C
            LD   (DE),A            ; 2 bytes

            CALL deadd5            ; pointer4
            CALL shift2            ; needs 4x shift right
            CALL shift2
            LD   (DE),A
            DEC  DE
            LD   A,C
            LD   (DE),A

            LD   A,18              ; point to next line
            ADD  A,E
            LD   E,A

            DEC  B                 ; decrease Y of ball
            JR   NC,brlp           ; do 4 lines

; now add ball
ballx       LD   A,0               ; fetch x-position
            RRCA
            RRCA
            RRCA                   ; div 8
            AND  31
            LD   DE,balldata
            ADD  A,E
            LD   E,A               ; point to ballposition
```

```
            LD   (erball+1),DE     ; save for erase
            LD   A,(ballx+1)       ; again get ball-x
            AND  7                 ; to read from
            LD   HL,nxtlin         ; preset shifted table
            JR   Z,bfound
            DEC  A
            ADD  A,A
            ADD  A,A
            LD   L,A
bfound      LD   C,1               ; shifted ball found
setball     LD   A,(DE)            ; get screenbyte
            XOR  (HL)              ; test against ball to set
            LD   B,A               ; result in B
            LD   A,(DE)            ; get screenbyte
            OR   (HL)              ; OR over ball
            LD   (DE),A            ; draw to screen
            SUB  B                 ; XOR vs OR, <>0 = collision
            LD   B,A               ; save in B
            INC  DE
            INC  HL
            LD   A,(DE)
            XOR  (HL)
            ADD  A,B               ; test col2 and col1 added
            LD   B,A
            LD   A,(DE)
            OR   (HL)
            LD   (DE),A
            SUB  B
            LD   A,C
            JR   NZ,collide        ; NZ means collision

            LD   A,L               ; point to next balludg
            ADD  A,C
            LD   L,A
            DEC  C                 ; next line other pointer
            DEC  C
            LD   A,E
            ADD  A,29
            LD   E,A
            JR   NC,setball        ; do full ball screen
            JR   nobrick

collide     ADD  A,5               ; from linemarker
            RRCA                   ; to linenr
            XOR  3
            LD   B,A
            LD   HL,dxy+1          ; collision means
            XOR  A                 ; swap direction
            LD   C,A               ; "cheap" LD C,0
            SUB  (HL)              ; dx = 0 - dx
            LD   (HL),A            ; save new dx
            LD   A,(ballx+1)       ; find which column is hit
            SUB  110
            JR   C,nobrick         ; we hit the bat
findc       SUB  35
            INC  C                 ; point to next column
            JR   NC,findc          ; find column
            LD   A,(bally+1)       ; BALLY
            SUB  B                 ; - line number
            LD   B,A               ; = pointer to right brick
            CALL brickfield        ; calculate brickfield
            DEC  (HL)              ; take of 1 layer
            LD   HL,sc+3           ; score 1 point
            CALL addscore          ; and score a point
```

```
nobrick    LD   DE,(baty+1)       ; D = baty
           LD   D,E               ; E = baty-copy

           LD   A,(lastk)
           SUB  %01111111
           JR   NZ,nospace

; space is pressed
           LD   HL,dxy+1
           OR   (HL)              ; change only when not set
           JR   NZ,nospace        ; only before start
           INC  A
           LD   (HL),A            ; set dx to 1
nospace    CP   %11111011-127     ; 127 from sub above
           JR   NZ,testdown
           INC  D                 ; do move UP
testdown   CP   %11111101-127
           JR   NZ,noplay
           DEC  D                 ; do move DOWN
noplay     LD   HL,bally+1
           LD   A,D
           SUB  29
           CP   148
; a preset dy to a direction after loading
dxy        LD   BC,256            ; get dx and dy of ball
           JR   NC,ballmove       ; test bat in range

           LD   A,C
           OR   A                 ; test dx

           LD   A,D
           LD   (baty+1),A        ; save new

           JR   NZ,ballmove       ; move ball when released

           SUB  E                 ; calculate dy

           JR   Z,delay           ; do NOT set dy to 0

           DEFB #CA               ; JP Z
newy       XOR  A                 ; dy = 0 - dy
           SUB  B

; set ball dy so it will move when released
           LD   (dxy+2),A         ; set dy bat as dy ball
           ADD  A,(HL)
           LD   (HL),A            ; do dy as bat
           JR   delay

ballmove   LD   A,B               ; dy to A
           ADD  A,(HL)            ; Y=Y+dy
           LD   (HL),A            ; save result
           SUB  5
           CP   171
           JR   NC,newy           ; test out of range

dodx       LD   HL,ballx+1
           LD   A,C               ; dx to A
           ADD  A,(HL)            ; X=X+dx
           LD   (HL),A
           CP   11
           JP   C,dead            ; missed the bat, live lost
           CP   232
```

```
           JR   C,delay           ; on screen

; right out of screen
           LD   DE,scrstack
           LD   B,44              ; we had 44 bricks
addbonus   LD   A,(DE)            ; fetch brick
           SUB  b0                ; calculate remainings
           LD   C,A
           JR   Z,bbonus          ; nothing left, next brick
bonus      LD   HL,sc+2           ; score 10 points
           CALL addscore          ; each layer = 10 points
           DEC  C
           JR   NZ,bonus
bbonus     INC  DE                ; check all bricks
           DJNZ addbonus
           JP   nextlev           ; go to next level

delay      XOR  A
           LD   B,A
wfr        LD   A,(HL)
           DJNZ wfr
           JP   erball            ; play on

scrstack   DEFB b1,b1,b1,b1

; balltable saved in scrstack
; for init only
ball0      DEFB %01100000,%00000000 ; b2
           DEFB %10010000,%00000000

           DEFB %00110000,%00000000 ; b3
           DEFB %01001000,%00000000

           DEFB %00011000,%00000000 ; b4
           DEFB %00100100

; After loading SCRSTACK will ALWAYS have values
; from #81 to #87. These values can be used as
; displaybuffer for an inverted line.
; So the top and bottom line can be displayed with
; displaybuffer from the ball as data and this part
; of the SCRSTACK as displaybuffer when at the end
; a RET is added.

edge       DEFB 0

           DEFB %00001100,%00000000 ; b5
           DEFB %00010010,%00000000

           DEFB %00000110,%00000000 ; b6
           DEFB %00001001,%00000000

           DEFB %00000011,%00000000
           DEFB %00000100,%10000000

           DEFB %00000001,%10000000
           DEFB %00000010,%01000000

           DEFB %00000000,%11000000
           DEFB %00000001,%00100000

           DEFB b1,b2,b3,b4
           DEFB b1,b1,b1,b1
```

```
           RET                    ; the return from EDGE-display

shift2     RR   C                 ; shift a brick 2 bits
           RRA
           RR   C
           RRA
           RET

deadd5     INC  DE
deadd4     INC  DE                ; incresase brickpointer
deadd3     INC  DE
           INC  DE
           INC  DE
getdata    PUSH HL                ; save brickpointer
           LD   L,(HL)            ; get datapointer
           LD   H,#43
           LD   C,(HL)            ; get data
           XOR  A                 ; reset for shift
           POP  HL                ; get brickpointer
           INC  HL                ; point to next
           RET

ten        LD   (HL),28
           DEC  HL
addscore   INC  (HL)
           LD   A,(HL)
           CP   38
           JR   Z,ten
           RET

brickfield LD   A,176             ; calculate brickpointer
           SUB  B
           RRA
           RRA
           AND  #3C
           ADD  A,C
           LD   HL,scrstack-1
           ADD  A,L
           LD   L,A
           RET

; Y pos of BAT wil be set here
batdata    DEFB 255
           DEFB 115,114,113,112,111,110,109
           DEFB 108,107,106,105,104,103,102,101,100
           DEFB 99,98,97,96,95,94,93,92,91,90,0,0,0
           DEFB 255

; the stack for the display of bricks
n          EQU  27
x          EQU  101

dfile      DEFB 118
sc         DEFB 28,28,28,28,0,0,0,0,0,0,0
           DEFB "B"+x,"R"+x,"E"+x,"A"+x,"K"+x
           DEFB "O"+x,"U"+x,"T"+x,0,0,0
lives      DEFB 31,0,0

hi         DEFB 28,28,28,28
           DEFB 118

; this is the STACK
space      EQU  #4380-$-#22-4
```

```
           DEFS space

highball   LD   R,A
           ADD  A,30
; here line is displaybuffer, but elswehere
; line is data for top/bottom lines.
line       DEFW 0,0,0,0,0,0,0,0,0,0
           DEFW 0,0,0,0
           DEFB 0
           JP   M,balllow         ; NR=neg, end of balldata
           JP   ballback          ; out of highmem

; the data for the visible bricks in each level
br0        DEFB 0
br1        DEFB %00100000
br2        DEFB %00110000
br3        DEFB %00111000
br4        DEFB %00111100
br5        DEFB %00111110
br6        DEFB %00111111

; here starts the ballscreen and init is done
; this part is reused in setting up the ballscreen
balldata   DEFB 0

init       LD   IX,hr
           LD   SP,highball
           LD   H,#3F             ; repair 48K bug
           LD   D,#BF
           LD   E,L               ; start at #3fnn
           LD   B,5               ; final bug at #43nn
           LDIR                   ; copy more than programsize

           LD   DE,nxtlin         ; set part of table on sysvar
           LD   HL,ball0
           LD   C,4
           LDIR

           LD   C,30              ; set rest of table on sysvar
           LD   DE,#4000
           LDIR

           LD   HL,scrstack       ; repair used screen
           LD   DE,scrstack+1
           LD   C,40
           LDIR

           LD   HL,balldata       ; clear the whole
           LD   DE,balldata+1     ; ballscreen only once
           LD   C,120

           JP   start-2           ; start the game with LDIR

vars       DEFB 128
last       EQU  $
```