Utilities are not included here.

```
; Quest for Cash
; --------------
; 2006, 2007 Hannu Nuotio

; Quest for Cash is a remake of the crap QBasic game of the same name (and
author).

; Start of project: 5.10.2006
; v.0.2 - 10.3.2007
; v.0.1 - 9.3.2007
; v.0.0.9 - 8.3.2007
; v.0.0.8 - 6.3.2007
; v.0.0.7 - 25.11.2006
; v.0.0.6 - 23.11.2006
; v.0.0.5 - 21.11.2006
; v.0.0.4 - 20.10.2006
; v.0.0.3 - 11.10.2006
; v.0.0.2 - 9.10.2006
; v.0.0.1 - 5.10.2006

; Compiles with ACME 0.91
; # acme --cpu 6502 -f cbm -o qfc.prg qfc.a

; Type SYS 3072 to start or use crunched version.

; Known bugs:

; TODO:

; Memory map:
; $0000-$00ff : temp zero page variables
; $0100-$03ff : unused
; $0400-$07e7 : screen memory
; $0800-$0bff : font memory
; $0c00-$xxxx : code
; $xxxx-$yyyy : variables (reused characters)
; $yyyy-$kkkk : color lookup table
; $kkkk-$zzzz : strings
; $zzzz-$jjjj : compressed levels
; $d800-$dbe7 : color memory
; $zzzz-$ffff : unused

; Notes:

; colors:
; 0 = black
; 1 = white
; 2 = red
; 3 = cyan
```

```
; 4 = purple
; 5 = green
; 6 = blue
; 7 = yellow
; 8 = orange
; 9 = brown
; 10 = lred
; 11 = dgray
; 12 = gray
; 13 = lgreen
; 14 = lblue
; 15 = lgrey


; --- Constants

!ct scr         ; C64 screencode

levels = 15     ; the amount of levels

joystick = 0        ; joystick: 0 = port 2, 1 = userport
easymode = 0        ; easier game (grid and eleb visible)

; - hw addresses
!if joystick = 0 {
joyport = $dc00     ; joystick port 2
joypddr = $dc02     ; joystick port 2 data direction register
} else {
joyport = $dd01     ; userport
joypddr = $dd03     ; userport data direction register
}
screen = $0400      ; screen address
color =  $d800      ; color ram address
scrtocol = $d4      ; screen->color difference MSB
vicborder = $d020       ; border color register
vicbackgnd = $d021      ; background color register
vicraster = $d012       ; raster compare register

; - input consts
joyrepeat = 20          ; joystick repeat delay
rastercmp = 200         ; raster line to wait for

; - tiles
tground = $40
twall = $41
tswall = $42
tdoor1 = $43
tdoor2 = $44
tsdoor = $45
tkey1 = $46
tkey2 = $47
```

```
tskey = $48
tbomb = $49
tlbomb = $4a
tunused1 = $4b
tunused2 = $4c
trock = $4d
trockw = $4e
trockh = $4f
thole = $50
twater = $51
tplank = $52
tplankw = $53
telea = $54
teleb = $55
teles = $56
ttele = $57
tzapd = $58
tzapu = $59
tzapr = $5a
tzapl = $5b
tcash = $5c
tplayer = $5d
tdeath = $5e
tfire = $5f

; - text locations
gamescreenloc = screen+40*2
gamescreenendloc = gamescreenloc+40*20
gamecolorloc = color+40*2
statustextloc = screen+40*24
statusbartext1loc = screen+40*23+1
statusbartext2loc = screen+40*23+29
bcdbomb = statusbartext1loc+2
bcdplank = statusbartext1loc+7
bcdkey1 = statusbartext1loc+12
bcdkey2 = statusbartext1loc+17
bcdskey = statusbartext1loc+22
bcdlevel = statusbartext2loc+6
helptextlen = 40*20+1
pickuptextloc = pickuptext+24
useitemtextloc = useitemtext+19
useitemtextloc2 = useitemtext+24

; - text color
gametitletextcol = 3
helptextcol = 15
statusbartext1col = 1
statusbartext2col = 2
restarttextcol = 10
nextleveltextcol = 13
zaptextcol = 7
```

```
gamewontextcol = 14
pickuptextcol = 12
useitemtextcol = 12
dropbombtextcol = 5
detonatebombtextcol = 8
teleporttextcol = 3
electextcol = 6


; --- Variables

; - zero page

tmpptr = $39        ; temporary zp pointer
scrptr = $fb        ; zp pointer to screen
colptr = $fd        ; zp pointer to color
tmpvar = $ff        ; temporary zp variable

; - reused chars

tmp = Chars

; last joystick state
lastjoy = Chars+1

; joystick repeat counter
joycount = Chars+2

; interrupt state
intstate = Chars+3

; pointer storage
storedptr = Chars+4
storedptrh = Chars+5
ccurlevel = Chars+6
ccurlevelh = Chars+7
cnextlevel = Chars+8
cnextlevelh = Chars+9

; current/selected level
level = Chars+10

; items
ibomb = Chars+11
iplank = Chars+12
ikey1 = Chars+13
ikey2 = Chars+14
iskey = Chars+15

; game state
playerptr = Chars+16
```

```
playerptrh = Chars+17
underplayer = Chars+18
lbomb = Chars+19
underlbomb = Chars+20
underlbombu = Chars+21
underlbombd = Chars+22
underlbombl = Chars+23
underlbombr = Chars+24
lbombptr = Chars+25
lbombptrh = Chars+26
movedir = Chars+27
newptr = Chars+28
newptrh = Chars+29


; --- Main

; start of program
*=$0c00
mlcodeentry:

; - interrupt setup
; from "An Introduction to Programming C-64 Demos" by Puterman aka Linus
Åkerlund
; http://user.tninet.se/~uxm165t/demo_programming/demo_prog/demo_prog.html
; ... + modifications
;
sei     ; interrupts off
lda #$7f
ldx #$01
sta $dc0d   ; Turn off CIA 1 interrupts
sta $dd0d   ; Turn off CIA 2 interrupts
stx $d01a   ; Turn on raster interrupts
lda #<int    ; low part of address of interrupt handler code
ldx #>int    ; high part of address of interrupt handler code
ldy #250    ; line to trigger interrupt
sta $0314   ; store in interrupt vector
stx $0315
sty $d012
lda #<nmi    ; low part of address of NMI handler code
ldx #>nmi    ; high part of address of NMI handler code
sta $0318   ; store in NMI vector
stx $0319
lda #0
sta intstate    ; set interrupt state to 0
lda $dc0d   ; ACK CIA 1 interrupts
lda $dd0d   ; ACK CIA 2 interrupts
asl $d019   ; ACK VIC interrupts
cli     ; interrupts on
```

```
; disable bcd-mode
cld

; set joyport to input
lda #0
sta joypddr

; copy character rom to $0800-$09ff
; copy custom chars to  $0a00-$0bff
ldx #0      ; 255 loops
sei     ; interrups off
lda $1
and #$fb
sta $1      ; character rom on
-   lda $d000,x    ; load from char-rom
    sta $0800,x    ; store to ram
    lda $d100,x    ; load from char-rom
    sta $0900,x    ; store to ram
    lda Chars,x    ; load from custom chars
    sta $0a00,x    ; store to ram
    lda Chars+$100,x; load from custom chars
    sta $0b00,x    ; store to ram
    inx
    bne -
lda $1
ora #$04
sta $1      ; character rom off
asl $d019   ; ACK VIC interrupts
cli     ; interrupts on

; font = $0800
lda $d018
and #$f1
ora #$02
sta $d018

; set colors
ldx #tswall
lda Colortable,x
sta vicborder   ; set border
lda #0
sta vicbackgnd  ; set background black

; print text
lda #>gametitletext
sta tmpptr+1
lda #<gametitletext
sta tmpptr
lda #>screen
sta scrptr+1
lda #<screen
```

```
sta scrptr
lda #gametitletextcol
jsr printstring

lda #>topbottomborder
sta tmpptr+1
lda #<topbottomborder
sta tmpptr
lda #>screen+40
sta scrptr+1
lda #<screen+40
sta scrptr
jsr printstring

lda #>screen+40*22
sta scrptr+1
lda #<screen+40*22
sta scrptr
jsr printstring

reloadstartmenu:
lda #>statusbartext1
sta tmpptr+1
lda #<statusbartext1
sta tmpptr
lda #>statusbartext1loc
sta scrptr+1
lda #<statusbartext1loc
sta scrptr
lda #statusbartext1col
jsr printstring

lda #>statusbartext2
sta tmpptr+1
lda #<statusbartext2
sta tmpptr
lda #>statusbartext2loc
sta scrptr+1
lda #<statusbartext2loc
sta scrptr
lda #statusbartext2col
jsr printstring

; - Start menu
;
startmenu:
lda #1
sta level
jsr erasestatustext

; set pointer to helptext page 1
```

```
startmenu_helppage1:
lda #>helptext
sta storedptr+1
lda #<helptext
sta storedptr

startmenu_drawhelp:
; check if valid helptext page
lda storedptr+1
cmp #>helptext
bcc startmenu_helppage1
cmp #>helptextlast
beq +
bcs startmenu_helppage1

; draw help screen
+ lda storedptr+1
sta tmpptr+1
lda storedptr
sta tmpptr
lda #>gamescreenloc
sta scrptr+1
lda #<gamescreenloc
sta scrptr
lda #helptextcol
jsr printstring

; start menu input loop
- jsr getinput
sta tmp
lda #%00000001  ; up
bit tmp     ; test if up
beq ++      ; skip if not
; up pressed
lda storedptr
sec
sbc #<helptextlen
sta storedptr
lda storedptr+1
sbc #>helptextlen
sta storedptr+1
jmp startmenu_drawhelp
++ lda #%00000010   ; down
bit tmp     ; test if down
beq ++      ; skip if not
; down pressed
lda storedptr
clc
adc #<helptextlen
sta storedptr
lda storedptr+1
```

```
adc #>helptextlen
sta storedptr+1
jmp startmenu_drawhelp
++ lda #%00000100   ; left
bit tmp     ; test if left
beq ++      ; skip if not
; left pressed
dec level
bne +
inc level
bne -
+ lda #>bcdlevel
sta tmpptr+1
lda #<bcdlevel
sta tmpptr
jsr bcddec
jmp -
++ lda #%00001000   ; right
bit tmp     ; test if right
beq ++      ; skip if not
; right pressed
inc level
lda #levels+1
cmp level
bne +
dec level
bne -
+ lda #>bcdlevel
sta tmpptr+1
lda #<bcdlevel
sta tmpptr
jsr bcdinc
jmp -
++ lda #%00010000   ; fire
bit tmp     ; test if fire
bne +       ; jump if pressed
jmp -       ; back to input loop


; init game
+ lda #>clevels
sta tmpptr+1
lda #<clevels
sta tmpptr
lda level
sta tmp

; load level (selected, next or restarted)
loadlevel:
jsr initlevel
```

```
; find starting position
lda #tplayer
jsr findchar
lda tmpptr
sta playerptr
lda tmpptr+1
sta playerptrh
lda #tground
sta underplayer

; - Game loop
;
gameloop:
; get input
jsr getinput
sta movedir
jsr erasestatustext
lda #%00000001  ; up
bit movedir ; test if up
beq +       ; skip if not
; up pressed
sta movedir ; eliminate possible diagonals
lda #>-40
sta storedptrh
lda #<-40
sta storedptr
jmp movement
+ lda #%00000010    ; down
bit movedir ; test if down
beq +       ; skip if not
; down pressed
sta movedir ; eliminate possible diagonals
lda #>40
sta storedptrh
lda #<40
sta storedptr
jmp movement
+ lda #%00000100    ; left
bit movedir ; test if left
beq +       ; skip if not
; left pressed
sta movedir ; eliminate possible diagonals
lda #>-1
sta storedptrh
lda #<-1
sta storedptr
jmp movement
+ lda #%00001000    ; right
bit movedir ; test if right
beq +       ; skip if not
; right pressed
```

```
sta movedir ; eliminate possible diagonals
lda #>1
sta storedptrh
lda #<1
sta storedptr
jmp movement
+ lda #%00010000    ; fire
bit movedir ; test if fire
beq gameloop    ; jump back if not
; fire pressed
lda lbomb   ; check if live bomb
beq +
; explode bomb
jmp explodebomb
+ lda ibomb ; check if any bombs
beq ++
; drop bomb
jmp dropbomb
; restart?
++ lda #>restarttext
sta tmpptr+1
lda #<restarttext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #restarttextcol
jsr printstring
jsr getinput
sta movedir
jsr erasestatustext
lda #%00010000  ; fire
bit movedir ; test if fire
bne +       ; skip if is
jmp gameloop
+ lda ccurlevel
sta tmpptr
lda ccurlevelh
sta tmpptr+1    ; tmpptr -> current level
lda #1
sta tmp     ; load 1 level
jmp loadlevel   ; restart level

movement:
; calculate new location
clc
lda playerptr
adc storedptr
sta newptr
sta tmpptr
```

```
lda playerptrh
adc storedptrh
sta tmpptr+1
sta newptrh ; tmpptr, newptr -> new location

; check left<->right wrap
lda #%00001100  ; left or right
bit movedir ; test if left or right
beq movement_check  ; skip if not
ldx newptrh
lda newptr
jsr mod40
sta tmp     ; tmp = new location mod 40
lda #%00000100  ; left
bit movedir ; test if left
beq +       ; skip if right
lda #39
cmp tmp     ; check if 39 (left->right)
bne movement_check  ; skip if not
jmp gameloop    ; left->right wrap, back to loop
+ lda tmp   ; check if 0 (right->left)
bne movement_check  ; skip if not
jmp gameloop    ; right->left wrap, back to loop

; check if movement ok
movement_check:
ldy #0
lda (tmpptr),y
tax     ; x = tile in new position

; items
lda #tground
cpx #tkey1  ; check if key1
bne +       ; if not, skip
sta (tmpptr),y  ; erase item
inc ikey1
lda #>bcdkey1
sta tmpptr+1
lda #<bcdkey1
sta tmpptr
jsr bcdinc  ; increase count
jsr printpickup
jmp movement_draw
+ cpx #tkey2    ; check if key2
bne +       ; if not, skip
sta (tmpptr),y  ; erase item
inc ikey2
lda #>bcdkey2
sta tmpptr+1
lda #<bcdkey2
sta tmpptr
```

```
jsr bcdinc  ; increase count
jsr printpickup
jmp movement_draw
+ cpx #tskey    ; check if skey
bne +       ; if not, skip
sta (tmpptr),y  ; erase item
inc iskey
lda #>bcdskey
sta tmpptr+1
lda #<bcdskey
sta tmpptr
jsr bcdinc  ; increase count
jsr printpickup
jmp movement_draw
+ cpx #tbomb    ; check if bomb
bne +       ; if not, skip
sta (tmpptr),y  ; erase item
inc ibomb
lda #>bcdbomb
sta tmpptr+1
lda #<bcdbomb
sta tmpptr
jsr bcdinc  ; increase count
jsr printpickup
jmp movement_draw
+ cpx #tplank   ; check if plank
bne +       ; if not, skip
sta (tmpptr),y  ; erase item
inc iplank
lda #>bcdplank
sta tmpptr+1
lda #<bcdplank
sta tmpptr
jsr bcdinc  ; increase count
jsr printpickup
jmp movement_draw

; item usage
+ cpx #twater   ; check if water
bne ++      ; if not, skip
lda iplank  ; check if any planks
bne +       ; skip if is
jmp gameloop    ; back to gameloop
+ lda #tplankw
sta (tmpptr),y  ; water->plank on water
dec iplank
lda #>bcdplank
sta tmpptr+1
lda #<bcdplank
sta tmpptr
jsr bcddec  ; decrease count
```

```
lda #tplank
jsr printuseitem
jmp movement_draw
++ cpx #tdoor1  ; check if door1
bne ++      ; if not, skip
lda ikey1   ; check if any key1's
bne +       ; skip if is
jmp gameloop    ; back to gameloop
+ lda #tground
sta (tmpptr),y  ; door->ground
dec ikey1
lda #>bcdkey1
sta tmpptr+1
lda #<bcdkey1
sta tmpptr
jsr bcddec  ; decrease count
lda #tkey1
jsr printuseitem
jmp movement_draw
++ cpx #tdoor2  ; check if door2
bne ++      ; if not, skip
lda ikey2   ; check if any key2's
bne +       ; skip if is
jmp gameloop    ; back to gameloop
+ lda #tground
sta (tmpptr),y  ; door->ground
dec ikey2
lda #>bcdkey2
sta tmpptr+1
lda #<bcdkey2
sta tmpptr
jsr bcddec  ; decrease count
lda #tkey2
jsr printuseitem
jmp movement_draw
++ cpx #tsdoor  ; check if sdoor
bne ++      ; if not, skip
lda iskey   ; check if any skeys
bne +       ; skip if is
jmp gameloop    ; back to gameloop
+ lda #tground
sta (tmpptr),y  ; door->ground
dec iskey
lda #>bcdskey
sta tmpptr+1
lda #<bcdskey
sta tmpptr
jsr bcddec  ; decrease count
lda #tskey
jsr printuseitem
jmp movement_draw
```

```
; special stuff
++ cpx #teles   ; check if elec. switch
bne ++      ; if not, skip
jmp elecswitch  ; switch elec.
++ cpx #tcash   ; check if cash
bne ++      ; if not, skip
jmp nextlevel
++ cpx #ttele   ; check if teleport
bne ++      ; if not, skip
jsr domovement  ; move into teleport (erase curr. tele)
lda #ttele  ; a = ttele
jsr findchar    ; tmpptr -> other teleport
lda tmpptr
sta newptr
lda tmpptr+1
sta newptrh ; newptr -> other teleport
lda #>teleporttext
sta tmpptr+1
lda #<teleporttext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #teleporttextcol
jsr printstring

jmp movement_draw   ; teleport

; rock
++ cpx #trock   ; check if rock
bne ++      ; if not, skip
jmp rockmove
++ cpx #trockw  ; check if rock on plank
bne ++      ; if not, skip
jmp rockmove

; simple movement
++ cpx #tground ; check if ground
beq movement_draw   ; if it is, move
cpx #tplankw    ; check if plank on water
beq movement_draw   ; if it is, move
cpx #thole  ; check if hole
beq movement_draw   ; if it is, move
cpx #teleb  ; check if inactive elec.
beq movement_draw   ; if it is, move
cpx #tlbomb ; check if inactive elec.
beq movement_draw   ; if it is, move
jmp gameloop    ; no movement, back to loop

movement_draw:
```

```
jsr domovement

; check zappers
zapcheck:
; check up
lda #>-40
sta tmpptr+1
lda #<-40
sta tmpptr
lda #tzapd
jsr zapsearch
dex
bne death
; check down
lda #>40
sta tmpptr+1
lda #<40
sta tmpptr
lda #tzapu
jsr zapsearch
dex
bne death
; check left
lda #>-1
sta tmpptr+1
lda #<-1
sta tmpptr
lda #tzapr
jsr zapsearch
dex
bne death
; check right
lda #>1
sta tmpptr+1
lda #<1
sta tmpptr
lda #tzapl
jsr zapsearch
dex
bne death
jmp gameloop    ; end of game loop

; - death
;
death:
lda playerptr
sta scrptr
lda playerptrh
sta scrptr+1
lda #tdeath
jsr printtile
```

```
lda #>zaptext
sta tmpptr+1
lda #<zaptext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #zaptextcol
jsr printstring
- jsr getinput
sta movedir
lda #%00010000  ; fire
bit movedir ; test if fire
beq -
lda ccurlevel
sta tmpptr
lda ccurlevelh
sta tmpptr+1    ; tmpptr -> current level
lda #1
sta tmp     ; load 1 level
jmp loadlevel   ; restart level

; - dropbomb
;
dropbomb:
lda #>dropbombtext
sta tmpptr+1
lda #<dropbombtext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #dropbombtextcol
jsr printstring
sec
lda playerptr
sbc #40
sta lbombptr
lda playerptrh
sta lbombptrh
bcs +
dec lbombptrh   ; lbombptr -> bomb_location - 40
+ dec ibomb
lda #>bcdbomb
sta tmpptr+1
lda #<bcdbomb
sta tmpptr
jsr bcddec
lda underplayer
```

```
cmp #teleb
bne +
lda #tground
+ sta underlbomb
lda #tlbomb
sta underplayer
inc lbomb
lda #0
sta underlbombl
sta underlbombr
jmp gameloop

; - elecswitch
;
elecswitch:
lda #>electext
sta tmpptr+1
lda #<electext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #electextcol
jsr printstring

; point to start of game screen
lda #>gamescreenloc
sta scrptr+1
lda #<gamescreenloc
sta scrptr
ldy #0
; replace telea<->teleb
- lda (scrptr),y
cmp #telea
bne +
lda #teleb
jsr printtile
jmp ++
+ cmp #teleb
bne ++
lda #telea
jsr printtile
++ inc scrptr
bne +
inc scrptr+1
+ lda #>gamescreenendloc
cmp scrptr+1
bne -
lda #<gamescreenendloc
cmp scrptr
```

```
bne -
lda underplayer
cmp #telea
bne +
lda #teleb
sta underplayer
bne ++
+ cmp #teleb
bne ++
lda #telea
sta underplayer
++ jmp zapcheck

; - rockmove
;
rockmove:
; calculate new rock location
clc
lda newptr
sta tmpptr
adc storedptr
sta scrptr
lda newptrh
sta tmpptr+1    ; tmpptr -> old rock location
adc storedptrh
sta scrptr+1    ; scrptr -> new rock location

; check for left & right edges
lda #%00001100  ; left or right
bit movedir ; test if left or right
beq ++      ; skip if not
ldx newptrh
lda newptr
jsr mod40
sta tmp     ; tmp = new location mod 40
lda #%00000100  ; left
bit movedir ; test if left
beq +       ; skip if right
lda #39
cmp tmp     ; check if 39 (left->right)
bne ++      ; skip if not
jmp gameloop    ; left->right wrap, back to loop
+ lda tmp   ; check if 0 (right->left)
bne ++      ; skip if not
jmp gameloop    ; right->left wrap, back to loop

; check if rock can be moved
++ ldy #0
lda (scrptr),y
tax
lda Rocktable,x
```

```
bne +       ; 0 -> cannot move
jmp gameloop

; draw new rock
+ jsr printtile
; erase old rock
lda newptr
sta scrptr
lda newptrh
sta scrptr+1
ldx #tground
lda (scrptr),y
cmp #trockw
bne +
ldx #tplankw
+ txa
jsr printtile
jmp movement_draw

; - explodebomb
;
explodebomb:
lda #>detonatebombtext
sta tmpptr+1
lda #<detonatebombtext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #detonatebombtextcol
jsr printstring

; erase player
lda playerptr
sta scrptr
lda playerptrh
sta scrptr+1
lda underplayer
jsr printtile

; check for left & right edges
lda lbombptr
sta tmpptr
ldx lbombptrh
stx tmpptr+1
jsr mod40
tax
cpx #39
beq ++
ldy #41
```

```
lda (tmpptr),y
sta underlbombr
++ cpx #0
beq ++
ldy #39
lda (tmpptr),y
sta underlbombl

; ignite flames, transform tiles
++ ldy #0   ; up
lda #tfire
jsr replacetile
tax
lda Bombtable,x ; transform old tile
sta underlbombu
ldy #80     ; down
lda #tfire
jsr replacetile
tax
lda Bombtable,x ; transform old tile
sta underlbombd
lda underlbombl ; left (if not on edge)
beq ++
ldy #39     ; left
lda #tfire
jsr replacetile
tax
lda Bombtable,x ; transform old tile
sta underlbombl
++ lda underlbombr  ; right (if not on edge)
beq ++
ldy #41     ; right
lda #tfire
jsr replacetile
tax
lda Bombtable,x ; transform old tile
sta underlbombr
++ ldy #40      ; center
lda #tfire
jsr replacetile
ldx underlbomb
lda Bombtable,x ; transform old tile
sta underlbomb

; artistic delay
++ ldx #17
jsr delay

; redraw tiles
ldy #0      ; up
lda underlbombu
```

```
jsr replacetile
ldy #80     ; down
lda underlbombd
jsr replacetile
lda underlbombl ; left (if not on edge)
beq ++
ldy #39     ; left
jsr replacetile
++ lda underlbombr  ; right (if not on edge)
beq ++
ldy #41     ; right
jsr replacetile
++ ldy #40      ; center
lda underlbomb
jsr replacetile

; redraw player
lda playerptr
sta scrptr
lda playerptrh
sta scrptr+1
ldy #0
lda (scrptr),y
sta underplayer
lda #tplayer
jsr printtile
dec lbomb
jmp zapcheck

; - nextlevel
;
nextlevel:
jsr domovement
lda level   ; check if last level
cmp #levels
bne +
jmp gamewon ; game won
+ inc level ; next level
lda #>nextleveltext
sta tmpptr+1
lda #<nextleveltext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #nextleveltextcol
jsr printstring
- jsr getinput
sta movedir
lda #%00010000  ; fire
```

```
bit movedir ; test if fire
beq -
lda #>bcdlevel
sta tmpptr+1
lda #<bcdlevel
sta tmpptr
jsr bcdinc  ; increase level
lda cnextlevel
sta tmpptr
lda cnextlevelh
sta tmpptr+1    ; tmpptr -> next level
lda #1
sta tmp     ; load 1 level
jmp loadlevel   ; load next level

; - gamewon
;
gamewon:
lda #>gamewontext
sta tmpptr+1
lda #<gamewontext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #gamewontextcol
jsr printstring
- jsr getinput
sta movedir
lda #%00010000  ; fire
bit movedir ; test if fire
beq -
jmp reloadstartmenu


; --- Interrupt routines

; - IRQ
;
int:
asl $d019   ; ACK interrupt (to re-enable it)
pla
tay
pla
tax
pla     ; pop y,x and a from stack
rti     ; return

; - NMI
;
```

```
nmi:
rti     ; return


; --- Subroutines

; - printpickup
; parameters:
;  x = item
;
printpickup:
stx pickuptextloc
lda #>pickuptext
sta tmpptr+1
lda #<pickuptext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #pickuptextcol
jsr printstring
rts

; - printuseitem
; parameters:
;  a = item
;  x = used on
;
printuseitem:
sta useitemtextloc
stx useitemtextloc2
lda #>useitemtext
sta tmpptr+1
lda #<useitemtext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
lda #useitemtextcol
jsr printstring
rts


; - zapsearch
; parameters:
;  playerptr -> player
;  tmpptr = direction (+-1, +-40)
;  a = fatal zapper tile
; returns:
```

```
;  x = 1 if ok, 0 if zapped
;  scrptr -> zapper
;
zapsearch:
ldy #0
sta tmpvar  ; tmpvar = fatal zapper tile
cmp #tzapl  ; check if left
bne +
ldy #2
+ cmp #tzapr    ; check if right
bne +
ldy #1
+ sty movedir   ; movedir = 1 if left, 2 if right
lda playerptr
sta scrptr
ldx playerptrh
stx scrptr+1    ; scrptr -> player


; move to next location
- clc
lda scrptr
adc tmpptr
sta scrptr
lda scrptr+1
adc tmpptr+1
sta scrptr+1

; check for left&right edges
ldy movedir
cpy #0
beq ++      ; skip if up or down
lda scrptr
ldx scrptr+1
jsr mod40
ldx #1
cmp #0
bne +
cpy #2
beq +++     ; moving right, left edge reached (x=1)
+ cmp #39
bne ++
cpy #1
beq +++     ; moving left, right edge reached (x=1)

; check if fatal zapper
++ ldx #0
ldy #0
lda (scrptr),y
cmp tmpvar
beq +++     ; jump if fatal zapper (x=0)
```

```
ldx #1
cmp #tground    ; check if ground
beq -
cmp #twater ; check if water
beq -
cmp #tplankw    ; check if plank on water
beq -
cmp #thole  ; check if hole
beq -
cmp #teleb  ; check if inactive elec.
beq -
+++ rts

; - replacetile
; parameters:
;  tmpptr+y -> location
;  a = new tile
; returns:
;  a = old tile
;
replacetile:
sta replacetilenew
lda (tmpptr),y  ; a = old tile
pha     ; to stack
lda tmpptr+1
sta scrptr+1
clc
tya     ; a = offset
adc tmpptr
sta scrptr
bcc ++
inc scrptr+1    ; scrptr = tmpptr+y
replacetilenew=*+1
++ lda #123
jsr printtile
pla     ; a = old tile
rts

; - delay
; parameters:
;  x = count
;
delay:
lda #rastercmp
-- cmp vicraster
bne --      ; wait until raster = rastercmp
- cmp vicraster
beq -       ; wait until raster != rastercmp
dex
bne --
rts
```

```
; - domovement
; parametes:
;  playerptr -> current player position
;  newptr -> new player position
;
domovement:
; redraw old position
lda playerptr
sta scrptr
lda playerptrh
sta scrptr+1
lda underplayer
jsr printtile
; move & draw
lda newptr
sta playerptr
sta scrptr
lda newptrh
sta playerptrh
sta scrptr+1
ldy #0
lda (scrptr),y
sta underplayer
lda #tplayer
jsr printtile
rts

; - erasestatustext
;
erasestatustext:
lda #>blanktext
sta tmpptr+1
lda #<blanktext
sta tmpptr
lda #>statustextloc
sta scrptr+1
lda #<statustextloc
sta scrptr
jsr printstring
rts

; - bcdinc
; parameters:
;  tmpptr -> BCD MSB
;
bcdinc:
ldy #1
- lda (tmpptr),y
clc
adc #1
cmp #'9'+1
```

```
bne +
lda #'0'
sta (tmpptr),y
dey
beq -
+ sta (tmpptr),y
rts

; - bcddec
; parameters:
;  tmpptr -> BCD MSB
;
bcddec:
ldy #1
- lda (tmpptr),y
sec
sbc #1
cmp #'0'-1
bne +
lda #'9'
sta (tmpptr),y
dey
beq -
+ sta (tmpptr),y
rts

; - initlevel
; parameters:
;  tmpptr -> level data
;  tmp = levels to advance
; returns:
;  ccurlevel -> current level data
;  cnextlevel -> next level data
;  ibomb, iplank ... = 0
;
initlevel:
; blank screen
lda $d011
and #$ef
sta $d011

; advance tmp levels
- lda tmpptr
sta ccurlevel
lda tmpptr+1
sta ccurlevelh  ; ccurlevel -> current level
jsr unclevel
dec tmp
bne -

lda tmpptr
```

```
sta cnextlevel
lda tmpptr+1
sta cnextlevelh ; cnextlevel -> next level

; reset items / status
jsr erasestatustext
lda #>statusbartext1
sta tmpptr+1
lda #<statusbartext1
sta tmpptr
lda #>statusbartext1loc
sta scrptr+1
lda #<statusbartext1loc
sta scrptr
lda #statusbartext1col
jsr printstring

lda #0
sta ibomb
sta iplank
sta ikey1
sta ikey2
sta iskey
sta lbomb

; unblank screen
lda $d011
ora #$10
sta $d011
rts

; - findchar
; parameters:
;  a = char to find
; returns:
;  tmpptr -> char at gamescreen
;
findchar:
sta tmp
lda #>gamescreenloc
sta tmpptr+1
lda #<gamescreenloc
sta tmpptr
ldy #0
- lda (tmpptr),y
cmp tmp     ; test if char
beq +       ; end if zero
inc tmpptr
bne -       ; loop if not zero
inc tmpptr+1
bne -
```

```
+ rts

; - mod40
; parameters:
;  a:x -> screen
; returns:
;  a = (a:x-screen_offset) mod 40
;
mod40:
dex
dex
dex
dex     ; a:x -= screen_offset
- cpx #0    ; test MSB
bne +       ; jump if > 0
cmp #40     ; test LSB
bcc ++      ; jump if < 40
+ sec
sbc #40     ; a -= 40
bcs -       ; jump if no borrow
dex     ; borrow
jmp -
++ rts      ; return

; - printtile
; parameters:
;  scrptr -> screen location to print to
;  a = the tile to print
;
printtile:
tax     ; put tile to x

; set colptr according to scrptr
lda scrptr
sta colptr
lda scrptr+1
clc
adc #scrtocol
sta colptr+1

; print tile
ldy #0
txa     ; get tile from x
sta (scrptr),y  ; print tile
lda Colortable,x    ; get tile color
sta (colptr),y  ; set color
rts     ; return

; - printstring
; parameters:
;  scrptr -> screen location to print to
```

```
;  tmpptr -> string to print
;  a = the color of normal text
;
printstring:
sta textcolor

; set colptr according to scrptr
lda scrptr
sta colptr
lda scrptr+1
clc
adc #scrtocol
sta colptr+1

; string loop
ldy #0
- lda (tmpptr),y
beq +++     ; end if zero
sta (scrptr),y  ; print char
tax
and #$40
bne ++      ; jump if tile
textcolor=*+1
lda #3      ; was just text
bne +
++ lda Colortable,x
+ sta (colptr),y    ; set color
iny
bne -       ; loop if not zero
inc tmpptr+1
inc scrptr+1
inc colptr+1
bne -
+++
rts     ; return

; - unclevel
; parameters:
;  tmpptr -> compressed level
; returns:
;  tmpptr -> next compressed level
;  scrptr -> screen line after level
;  colptr -> color line after level
;
unclevel:
; point to start of game screen
lda #>gamescreenloc
sta scrptr+1
lda #<gamescreenloc
sta scrptr
lda #>gamecolorloc
```

```
sta colptr+1
lda #<gamecolorloc
sta colptr

; main uncompress routine
ldy #0
--
lda (tmpptr),y
cmp #$ff        ; code for "end of level"
beq +++
and #$1f
clc
adc #$40
tax         ; x = char
lda (tmpptr),y
lsr
lsr
lsr
lsr
lsr
sta tmpvar
inc tmpvar      ; tmpvar = loops
- txa
sta (scrptr),y
lda Colortable,x
sta (colptr),y
inc scrptr
bne +
inc scrptr+1
+ inc colptr
bne +
inc colptr+1
+ dec tmpvar
bne -
inc tmpptr
bne +
inc tmpptr+1
+ lda #>gamescreenendloc
cmp scrptr+1
bne --
lda #<gamescreenendloc
cmp scrptr
bne --
rts         ; return

; $ff found, rest of level is ground
+++
ldx #tground        ; x = ground char
lda Colortable,x
sta tmpvar      ; tmpvar = color
- txa
```

```
sta (scrptr),y
lda tmpvar
sta (colptr),y
inc scrptr
bne +
inc scrptr+1
+ inc colptr
bne +
inc colptr+1
+ lda #>gamescreenendloc
cmp scrptr+1
bne -
lda #<gamescreenendloc
cmp scrptr
bne -
inc tmpptr
bne +
inc tmpptr+1
+ rts           ; return

; - getinput
; returns:
;  a = action (xxxfrldu, 1 = active)
;
getinput:
lda #0
sta joycount    ; reset joyrepeat counter
-- lda #rastercmp
ldy joycount
- cmp vicraster
bne -           ; wait until raster = rastercmp
iny         ; y++
cpy #joyrepeat  ; check if y >= joyrepeat
bcc getinput_j  ; if not, skip
ldy #0          ; reset joyrepeat counter
lda #$ff        ; reset lastjoy for repeat
sta lastjoy
getinput_j: ; handle keyboard
sty joycount    ; store joycount
lda joyport ; a = joy
tax     ; save to x
eor lastjoy ; a = joy xor lastjoy
and lastjoy ; a = a and lastjoy
stx lastjoy ; update lastjoy
and #$1f        ; mask + test if anything is pressed
beq --          ; if not, wait
rts     ; return (a = action)


; --- Characters ($40-$5F)
```

```
; ...reused as variables

Chars
    ; ground
    ;    76543210
    !by %........ ;0
    !by %........ ;1
    !by %........ ;2
    !by %...##... ;3
    !by %...##... ;4
    !by %........ ;5
    !by %........ ;6
    !by %........ ;7

    ; wall
    ;    76543210
    !by %###.###. ;0
    !by %........ ;1
    !by %.###.### ;2
    !by %........ ;3
    !by %###.###. ;4
    !by %........ ;5
    !by %.###.### ;6
    !by %........ ;7

    ; super wall
    ;    76543210
    !by %######## ;0
    !by %######## ;1
    !by %######## ;2
    !by %######## ;3
    !by %######## ;4
    !by %######## ;5
    !by %######## ;6
    !by %######## ;7

    ; door1
    ;    76543210
    !by %...##... ;0
    !by %..####.. ;1
    !by %.######. ;2
    !by %.######. ;3
    !by %.####.#. ;4
    !by %.######. ;5
    !by %.######. ;6
    !by %.######. ;7

    ; door2
    ;    76543210
    !by %...##... ;0
    !by %..####.. ;1
```

```
    !by %.######. ;2
    !by %.######. ;3
    !by %.####.#. ;4
    !by %.######. ;5
    !by %.######. ;6
    !by %.######. ;7

    ; super door
    ;    76543210
    !by %...##... ;0
    !by %..####.. ;1
    !by %.######. ;2
    !by %.######. ;3
    !by %.####.#. ;4
    !by %.######. ;5
    !by %.######. ;6
    !by %.######. ;7

    ; key1
    ;    76543210
    !by %........ ;0
    !by %...##... ;1
    !by %..#..#.. ;2
    !by %...##... ;3
    !by %...#.... ;4
    !by %...#.... ;5
    !by %...###.. ;6
    !by %........ ;7

    ; key2
    ;    76543210
    !by %........ ;0
    !by %...##... ;1
    !by %..#..#.. ;2
    !by %...##... ;3
    !by %...#.... ;4
    !by %...#.... ;5
    !by %...###.. ;6
    !by %........ ;7

    ; super key
    ;    76543210
    !by %........ ;0
    !by %...##... ;1
    !by %..#..#.. ;2
    !by %...##... ;3
    !by %...#.... ;4
    !by %...#.... ;5
    !by %...###.. ;6
    !by %........ ;7
```

```
; bomb
;    76543210
!by %........ ;0
!by %....##.. ;1
!by %...#.... ;2
!by %..###... ;3
!by %.#####.. ;4
!by %.#####.. ;5
!by %..###... ;6
!by %........ ;7

; live bomb
;    76543210
!by %........ ;0
!by %....##.. ;1
!by %...#.... ;2
!by %..###... ;3
!by %.#####.. ;4
!by %.#####.. ;5
!by %..###... ;6
!by %........ ;7

; unused1
;    76543210
!by %#....... ;0
!by %........ ;1
!by %........ ;2
!by %........ ;3
!by %........ ;4
!by %........ ;5
!by %........ ;6
!by %........ ;7

; unused2
;    76543210
!by %##...... ;0
!by %........ ;1
!by %........ ;2
!by %........ ;3
!by %........ ;4
!by %........ ;5
!by %........ ;6
!by %........ ;7

; rock
;    76543210
!by %........ ;0
!by %...##... ;1
!by %..####.. ;2
!by %.######. ;3
!by %.######. ;4
```

```
    !by %.######. ;5
    !by %..####.. ;6
    !by %........ ;7

    ; rock+plank+water
    ;    76543210
    !by %........ ;0
    !by %...##... ;1
    !by %..####.. ;2
    !by %.######. ;3
    !by %.######. ;4
    !by %.######. ;5
    !by %..####.. ;6
    !by %........ ;7

    ; rock in a hole
    ;    76543210
    !by %........ ;0
    !by %........ ;1
    !by %........ ;2
    !by %...##... ;3
    !by %..####.. ;4
    !by %.######. ;5
    !by %.######. ;6
    !by %........ ;7

    ; hole
    ;    76543210
    !by %........ ;0
    !by %........ ;1
    !by %........ ;2
    !by %........ ;3
    !by %........ ;4
    !by %.######. ;5
    !by %#......# ;6
    !by %.######. ;7

    ; water
    ;    76543210
    !by %........ ;2
    !by %...##..# ;0
    !by %.##..##. ;1
    !by %........ ;2
    !by %........ ;2
    !by %...##..# ;3
    !by %.##..##. ;4
    !by %........ ;5

    ; plank
    ;    76543210
    !by %........ ;0
```

```
!by %...##... ;1
!by %...##... ;2
!by %...##... ;3
!by %...##... ;4
!by %...##... ;5
!by %...##... ;6
!by %........ ;7

; plank on water
;    76543210
!by %........ ;0
!by %.##..... ;1
!by %..##.... ;2
!by %...##... ;3
!by %....##.. ;4
!by %.....##. ;5
!by %......## ;6
!by %........ ;7

; electricity A
;    76543210
!by %........ ;0
!by %......#. ;1
!by %....##.. ;2
!by %..##.... ;3
!by %.######. ;4
!by %....##.. ;5
!by %..##.... ;6
!by %.#...... ;7

; electricity B
;    76543210
!by %........ ;0
!by %......#. ;1
!by %....##.. ;2
!by %..##.... ;3
!by %.######. ;4
!by %....##.. ;5
!by %..##.... ;6
!by %.#...... ;7

; electricity switch
;    76543210
!by %.#.....# ;0
!by %#.#...#. ;1
!by %###..#.. ;2
!by %#.#.#... ;3
!by %...#.#.. ;4
!by %..#..##. ;5
!by %.#...#.# ;6
!by %.....##. ;7
```

```
; teleport
;    76543210
!by %..####.. ;0
!by %.##..##. ;1
!by %#.#..#.# ;2
!by %#.#..#.# ;3
!by %#.#..#.# ;4
!by %#.####.# ;5
!by %##....## ;6
!by %.######. ;7

; zapper-down
;    76543210
!by %........ ;0
!by %.######. ;1
!by %..####.. ;2
!by %...##... ;3
!by %...##... ;4
!by %...##... ;5
!by %...##... ;6
!by %........ ;7

; zapper-up
;    76543210
!by %........ ;0
!by %...##... ;1
!by %...##... ;2
!by %...##... ;3
!by %...##... ;4
!by %..####.. ;5
!by %.######. ;6
!by %........ ;7

; zapper-right
;    76543210
!by %........ ;0
!by %.#...... ;1
!by %.##..... ;2
!by %.######. ;3
!by %.######. ;4
!by %.##..... ;5
!by %.#...... ;6
!by %........ ;7

; zapper-left
;    76543210
!by %........ ;0
!by %......#. ;1
!by %.....##. ;2
!by %.######. ;3
!by %.######. ;4
```

```
    !by %.....##. ;5
    !by %......#. ;6
    !by %........ ;7

    ; cash
    ;    76543210
    !by %..#..#.. ;0
    !by %.######. ;1
    !by %#.#..#.# ;2
    !by %#.#..#.. ;3
    !by %.######. ;4
    !by %..#..#.# ;5
    !by %#.#..#.# ;6
    !by %.######. ;7

    ; player
    ;    76543210
    !by %...##... ;0
    !by %..####.. ;1
    !by %...##... ;2
    !by %.######. ;3
    !by %...##... ;4
    !by %...##... ;5
    !by %..#..#.. ;6
    !by %..#..#.. ;7

    ; death
    ;    76543210
    !by %...##... ;0
    !by %...##... ;1
    !by %.######. ;2
    !by %.######. ;3
    !by %...##... ;4
    !by %...##... ;5
    !by %...##... ;6
    !by %...##... ;7

    ; fire
    ;    76543210
    !by %........ ;0
    !by %.#...#.. ;1
    !by %..#.#... ;2
    !by %.#..#.#. ;3
    !by %..##.#.. ;4
    !by %.##.###. ;5
    !by %.#.#..#. ;6
    !by %..####.. ;7


; --- Color lookup table (with $40 offset)
```

```
Colortable = *-$40
!if easymode = 0 {
!by 0   ; ground (black,0)
} else {
!by 5   ; ground
}
!by 2   ; wall (red,2)
!by 11  ; super wall (dgray,11)
!by 9   ; door1 (brown,9)
!by 7   ; door2 (yellow,7)
!by 15  ; super door (lgray,15)
!by 9   ; key1 (brown,9)
!by 7   ; key2 (yellow,7)
!by 15  ; super key (lgray,15)
!by 12  ; bomb (gray,12)
!by 1   ; live bomb (white,1)
!by 2   ; unused1
!by 3   ; unused2
!by 11  ; rock (dgray,11)
!by 11  ; rock+plank+water (dgray,11)
!by 11  ; rock in a hole (dgray,11)
!by 11  ; hole (dgray,11)
!by 6   ; water (blue,6)
!by 9   ; plank (brown,9)
!by 9   ; plank on water (brown,9)
!by 7   ; electricity A (yellow,7)
!if easymode = 0 {
!by 0   ; electricity B (black,0)
} else {
!by 6   ; electricity B
}
!by 5   ; electricity switch (green,5)
!by 4   ; teleport (purple,4)
!by 10  ; zapper-down (pink,10)
!by 10  ; zapper-up (pink,10)
!by 10  ; zapper-right (pink,10)
!by 10  ; zapper-left (pink,10)
!by 7   ; cash (yellow,7)
!by 12  ; player (gray,12)
!by 12  ; death (gray,12)
!by 2   ; fire (red,2)


; --- Bomb transform lookup table (with $40 offset)

Bombtable = *-$40
!by tground ; ground
!by tground ; wall
!by tswall  ; super wall
!by tground ; door1
!by tground ; door2
```

```
!by tsdoor  ; super door
!by tkey1   ; key1
!by tkey2   ; key2
!by tskey   ; super key
!by tbomb   ; bomb
!by 0       ; live bomb
!by 0       ; unused1
!by 0       ; unused2
!by tground ; rock
!by tplankw ; rock+plank+water
!by thole   ; rock in a hole
!by thole   ; hole
!by twater  ; water
!by tplank  ; plank
!by tplankw ; plank on water
!by telea   ; electricity A
!by tground ; electricity B
!by teles   ; electricity switch
!by ttele   ; teleport
!by tground ; zapper-down
!by tground ; zapper-up
!by tground ; zapper-right
!by tground ; zapper-left
!by tcash   ; cash
!by tplayer ; player
!by 0       ; death
!by 0       ; fire


; --- Rock transform lookup table (with $40 offset)

Rocktable = *-$40
!by trock   ; ground
!by 0       ; wall
!by 0       ; super wall
!by 0       ; door1
!by 0       ; door2
!by 0       ; super door
!by 0       ; key1
!by 0       ; key2
!by 0       ; super key
!by 0       ; bomb
!by 0       ; live bomb
!by 0       ; unused1
!by 0       ; unused2
!by 0       ; rock
!by 0       ; rock+plank+water
!by 0       ; rock in a hole
!by trockh  ; hole
!by twater  ; water
!by 0       ; plank
```

```
!by trockw  ; plank on water
!by 0       ; electricity A
!by trock   ; electricity B
!by 0       ; electricity switch
!by 0       ; teleport
!by 0       ; zapper-down
!by 0       ; zapper-up
!by 0       ; zapper-right
!by 0       ; zapper-left
!by 0       ; cash
!by 0       ; player
!by 0       ; death
!by 0       ; fire


; --- Strings

; - Common strings

gametitletext
!tx thole,trockh,trock," quest for cash ",tcash," by hannu nuotio
",trock,trockh,thole,0
topbottomborder
!fi 40,tswall
!by 0
statusbartext1
!tx tbomb,"=00 "
!tx tplank,"=00 "
!tx tkey1,"=00 "
!tx tkey2,"=00 "
!tx tskey,"=00",0
statusbartext2
!tx "level 01",0

teleporttext
!tx "           ",ttele,"  teleported  ",ttele,"            "
!by 0

electext
!tx "        ",telea," electricity switched ",telea,"        "
!by 0

pickuptext
!tx "              picked up x               "
!by 0

useitemtext
!tx "              used x on y               "
!by 0

dropbombtext
```

```
!tx "   dropped ",tlbomb,". press fire to detonate    "
!by 0

detonatebombtext
!tx "             ",tfire,tfire,tfire," boom! ",tfire,tfire,tfire,"
"
!by 0

restarttext
!tx "      press fire again to restart       "
!by 0

nextleveltext
!tx "you got the ",tcash,"! press fire for next level"
!by 0

gamewontext
!tx "       ",tcash,tcash,tcash," congratulations! ",tcash,tcash,tcash,"
"
!by 0

zaptext
!tx "  ",tzapr," zapped! ",tzapl,"  press fire to restart    "
!by 0

; - Help screens

;    |---------0---------0---------0--------|
helptext
!tx " help screen 1/3 : controls            "
!tx "                                       "
!tx "                                       "
!if joystick = 0 {
!tx "   use joystick in port 2              "
} else {
!tx "   use joystick in userport            "
}
!tx "                                       "
!tx "                                       "
!tx "menu controls:     joystick            "
!tx "                                       "
!tx "  start game   -     fire              "
!tx " select level  -  left/right           "
!tx "  browse help  -    up/down            "
!tx "                                       "
!tx "                                       "
!tx "game controls:     joystick            "
!tx "                                       "
!tx "     move      - up/down/left/right     "
!tx " restart level -  fire (if ",tbomb,"=00)        "
!tx "   drop bomb   -  fire (if ",tbomb,">00)        "
```

```
!tx " detonate bomb -  fire (if ",tlbomb,")           "
blanktext
!tx "                                          "
!by 0

!tx " help screen 2/3 : instructions         "
!tx "                                          "
!tx "move ",tplayer," to ",tcash,".                              "
!tx "                                          "
!tx "collect ",tkey1,tkey2,tskey," to open ", tdoor1,tdoor2,tsdoor,".
"
!tx "                                          "
!tx "collect and use ",tplank," to transform ",twater," to ",tplankw,".  "
!tx "                                          "
!tx "collect ",tbomb,", drop it (",tlbomb,") and detonate (",tfire,") to
destroy "
!tx twall,tdoor1,tdoor2,trock,trockh,tzapu,tzapd,tzapl,tzapr,tplankw," and
inactive ",telea,".   "
!tx "                                          "
!tx "use ",teles," to toggle ",telea,".                       "
!tx "use ",ttele," to teleport to another ",ttele,".         "
!tx "                                          "
!tx "avoid ",tzapu,tzapd,tzapl,tzapr
!tx " or they will turn ",tplayer," to ",tdeath,".    "
!tx "hide behind anything except "
!tx twater,tplankw,thole
!tx ".        "
!tx "                                          "
!tx "push ",trock, " into ",twater," to get rid of ",trock,". "
!tx "pushing ", trock, "into ",thole," makes ",trockh, " (which cannot be
pushed)."
!tx "push ",trock," over inactive ",telea," to destroy the ",telea,"."
!by 0

helptextlast
!tx " help screen 3/3 : credits & info       "
!tx "                                          "
!tx "quest for cash is a c64 port of an old  "
!tx "qbasic game, both by hannu nuotio       "
!tx "                                          "
!tx "levelset info:                          "
!tx " levels 01-15 are from the original qfc "
!tx "  by hannu nuotio & friends             "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
```

```
!tx "                                          "
!tx "                                          "
!tx "                                          "
!tx "                                          "
!by 0


; --- Levels (compressed)

clevels
; level 01
!by $e2, $e2, $e2, $62, $80, $62, $20, $17, $1d, $00, $06, $00, $02, $80,
$11, $80
!by $02, $00, $16, $00, $0d, $00, $14, $20, $15, $60, $0d, $20, $05, $00,
$17, $02
!by $c0, $03, $40, $12, $00, $02, $20, $09, $20, $01, $40, $0d, $00, $15,
$00, $10
!by $14, $00, $02, $1a, $20, $08, $00, $02, $20, $02, $1c, $20, $e2, $e2,
$e2, $e2
!by $e2, $ff
; level 02
!by $e1, $41, $e0, $e0, $e0, $80, $01, $00, $0d, $c0, $01, $e0, $e0, $e0,
$80, $01
!by $00, $0d, $c0, $01, $e0, $e0, $e0, $80, $01, $12, $0d, $20, $2d, $40,
$01, $e0
!by $e0, $e0, $80, $01, $00, $11, $c0, $01, $e0, $e0, $e0, $80, $01, $00,
$01, $00
!by $0d, $20, $01, $20, $01, $e0, $e0, $e0, $80, $21, $1c, $11, $4d, $00,
$01, $00
!by $01, $e0, $e0, $e0, $80, $01, $4d, $00, $0d, $00, $0d, $41, $e0, $e0,
$e0, $80
!by $01, $2d, $40, $0d, $40, $01, $e0, $e0, $e0, $80, $01, $0d, $20, $1d,
$80, $01
!by $e0, $e0, $e0, $80, $e1, $41, $ff
; level 03
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $c0, $81, $e0, $e0, $e0, $e0, $40,
$01, $00
!by $16, $00, $01, $e0, $a0, $12, $e0, $e0, $60, $01, $4d, $01, $e0, $e0,
$e0, $e0
!by $80, $11, $e0, $e0, $00, $54, $e0, $40, $a1, $e0, $e0, $60, $14, $0d,
$14, $e0
!by $40, $01, $60, $01, $e0, $e0, $60, $54, $e0, $40, $03, $75, $01, $e0,
$e0, $e0
!by $e0, $20, $01, $55, $1c, $01, $e0, $e0, $e0, $80, $06, $00, $1b, $20,
$01, $60
!by $01, $e0, $e0, $e0, $e0, $20, $a1, $e0, $e0, $e0, $e0, $e0, $e0, $e0,
$e0, $e0
!by $e0, $e0, $e0, $e0, $e0, $80, $01, $20, $01, $00, $01, $20, $61, $40,
$81, $e0
!by $20, $1d, $e0, $00, $01, $00, $01, $20, $01, $20, $01, $40, $01, $20,
$01, $e0
```

```
!by $e0, $e0, $21, $40, $01, $20, $61, $40, $81, $e0, $e0, $60, $01, $00,
$01, $20
!by $01, $20, $01, $00, $01, $e0, $01, $e0, $e0, $60, $01, $00, $01, $20,
$01, $20
!by $01, $20, $01, $40, $81, $e0, $e0, $60, $01, $20, $01, $00, $01, $20,
$01, $20
!by $01, $ff
; level 04
!by $e0, $e0, $e0, $e0, $e0, $00, $1d, $e0, $e0, $e0, $e0, $e0, $e0, $e0,
$e0, $e0
!by $c0, $55, $40, $16, $e0, $e0, $e0, $e0, $00, $15, $0d, $15, $e0, $e0,
$e0, $e0
!by $80, $55, $e0, $e0, $e0, $e0, $80, $54, $e0, $e0, $e0, $e0, $80, $14,
$0d, $14
!by $e0, $e0, $e0, $e0, $80, $54, $e0, $e0, $e0, $e0, $e0, $e0, $c0, $19,
$e0, $e0
!by $e0, $e0, $20, $06, $40, $1b, $00, $1a, $e0, $e0, $e0, $e0, $a0, $18,
$e0, $e0
!by $e0, $e0, $e0, $e0, $a1, $e0, $e0, $e0, $e0, $20, $03, $00, $04, $00,
$1c, $01
!by $e0, $e0, $e0, $e0, $20, $a1, $e0, $e0, $e0, $00, $07, $ff
; level 05
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $80, $0d, $00, $2d,
$00, $0d
!by $00, $2d, $20, $2d, $e0, $e0, $e0, $40, $8d, $09, $ed, $e0, $e0, $e0,
$20, $8d
!by $11, $0d, $00, $0d, $20, $2d, $00, $0d, $20, $1b, $e0, $e0, $80, $2d,
$00, $0d
!by $00, $2d, $00, $0d, $00, $2d, $00, $4d, $e0, $e0, $e0, $00, $2d, $00,
$0d, $00
!by $0d, $00, $1d, $0d, $00, $0d, $18, $0d, $00, $2d, $e0, $e0, $c0, $4d,
$00, $0d
!by $00, $0d, $00, $0d, $00, $0d, $20, $0d, $00, $2d, $e0, $e0, $e0, $4d,
$00, $2d
!by $00, $0d, $00, $0d, $00, $0d, $00, $4d, $e0, $e0, $e0, $20, $2d, $00,
$ad, $00
!by $4d, $e0, $e0, $c0, $1a, $60, $0d, $40, $4d, $00, $0d, $e0, $e0, $e0,
$e0, $e0
!by $e0, $e0, $e0, $e0, $60, $01, $00, $01, $19, $e0, $e0, $e0, $e0, $60,
$01, $00
!by $01, $e0, $e0, $e0, $e0, $80, $01, $00, $01, $e0, $e0, $e0, $e0, $80,
$01, $20
!by $e1, $01, $00, $e1, $21, $e0, $e0, $00, $21, $e0, $1a, $00, $01, $06,
$01, $00
!by $1b, $20, $03, $1c, $21, $e0, $e0, $40, $e1, $21, $00, $e1, $21, $ff
; level 06
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $40, $1d, $a0, $0d,
$e0, $e0
!by $e0, $e0, $a0, $0d, $20, $1b, $c0, $95, $e0, $e0, $a0, $0d, $e0, $60,
$15, $40
!by $15, $e0, $e0, $0d, $40, $0d, $20, $0d, $00, $12, $00, $12, $c0, $95,
```

```
$e0, $e0
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $51, $c0, $01, $04,
$01, $e0
!by $e0, $e0, $20, $1a, $11, $00, $11, $c0, $01, $06, $01, $e0, $e0, $e0,
$40, $11
!by $16, $11, $c0, $41, $e0, $e0, $e0, $40, $51, $e0, $e0, $e0, $e0, $e0,
$e0, $00
!by $19, $e0, $e0, $e0, $e0, $e0, $20, $1a, $00, $07, $e0, $61, $e0, $e0,
$e0, $e0
!by $60, $03, $34, $01, $e0, $e0, $e0, $e0, $60, $01, $1c, $14, $01, $e0,
$e0, $e0
!by $e0, $60, $61, $ff
; level 07
!by $a0, $21, $00, $0d, $40, $0d, $00, $0d, $c0, $0d, $40, $01, $60, $f1,
$20, $1d
!by $40, $01, $06, $00, $0d, $00, $0d, $00, $0d, $40, $0d, $40, $2d, $00,
$0d, $00
!by $0d, $21, $60, $d1, $81, $00, $21, $00, $0d, $00, $0d, $40, $0d, $00,
$0d, $00
!by $0d, $00, $0d, $00, $0d, $00, $0d, $e0, $b1, $a0, $01, $00, $0d, $00,
$0d, $00
!by $61, $00, $e1, $a1, $60, $91, $a0, $c1, $20, $41, $e0, $20, $51, $80,
$71, $a0
!by $0d, $e0, $00, $0d, $40, $41, $a0, $31, $a0, $51, $21, $03, $e1, $41,
$00, $a1
!by $e0, $00, $11, $c0, $31, $12, $11, $0d, $01, $0d, $00, $0d, $00, $16,
$00, $0d
!by $00, $0d, $01, $00, $01, $e0, $a0, $11, $e0, $51, $00, $01, $ed, $21,
$00, $21
!by $e0, $80, $11, $e0, $11, $40, $01, $e0, $01, $40, $01, $e0, $80, $11,
$e0, $11
!by $40, $01, $e0, $01, $40, $01, $e0, $80, $11, $e0, $11, $20, $0d, $01,
$e0, $01
!by $20, $0d, $01, $e0, $80, $11, $c0, $31, $e0, $60, $21, $12, $21, $e0,
$80, $11
!by $a0, $51, $e0, $80, $41, $11, $e0, $60, $51, $60, $71, $c0, $b4, $41,
$31, $e0
!by $20, $f1, $71, $c0, $14, $95, $41, $f1, $f1, $f1, $c0, $14, $15, $60,
$41, $31
!by $a0, $31, $20, $d1, $81, $c0, $14, $15, $20, $07, $00, $41, $11, $a0,
$71, $20
!by $51, $00, $12, $00, $04, $14, $20, $01, $c0, $14, $15, $60, $41, $c0,
$11, $32
!by $11, $40, $11, $60, $01, $14, $1c, $00, $e1, $e1, $01, $a0, $b1, $c0,
$81
; level 08
!by $1d, $e0, $c0, $11, $e0, $e0, $00, $8d, $16, $80, $41, $00, $0d, $a0,
$11, $e0
!by $e0, $2d, $00, $4d, $a0, $01, $07, $01, $e0, $11, $e0, $e0, $4d, $e0,
$00, $01
!by $07, $01, $e0, $11, $e0, $c0, $0d, $00, $0d, $00, $0d, $00, $2d, $80,
```

```
$41, $e0
!by $11, $e0, $c0, $2d, $00, $0d, $00, $4d, $e0, $e0, $11, $e0, $e0, $6d,
$00, $2d
!by $e0, $c0, $51, $e0, $e0, $40, $0d, $00, $0d, $e0, $a0, $11, $15, $19,
$15, $11
!by $e0, $e0, $e0, $e0, $40, $11, $1b, $1c, $1a, $11, $e0, $e0, $e0, $e0,
$40, $11
!by $15, $18, $15, $11, $e0, $e0, $e0, $e0, $60, $51, $e0, $e0, $e0, $e0,
$e0, $e0
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0,
$e0, $e0
!by $c0, $61, $00, $0d, $e0, $20, $0d, $e0, $e0, $a0, $a1, $e0, $e0, $e0,
$e0, $20
!by $01, $00, $12, $00, $24, $e0, $e0, $e0, $e0, $20, $a1, $e0, $e0, $e0,
$c0, $54
!by $00, $61, $e0, $e0, $e0, $e0, $14, $29
; level 09
!by $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $e0, $c0, $1b, $a0, $e1, $e1,
$e1, $c1
!by $e0, $0d, $01, $60, $01, $e0, $e0, $60, $03, $40, $01, $06, $40, $1d,
$60, $03
!by $00, $06, $20, $03, $80, $e1, $e1, $4d, $01, $e0, $0d, $01, $60, $01,
$60, $81
!by $e0, $20, $21, $00, $0d, $00, $01, $00, $15, $1b, $a0, $01, $60, $01,
$40, $a1
!by $40, $95, $20, $21, $0d, $00, $0d, $01, $e0, $00, $21, $03, $41, $e0,
$04, $40
!by $35, $14, $35, $20, $21, $40, $01, $e0, $00, $01, $80, $01, $20, $a1,
$40, $15
!by $14, $1c, $14, $15, $20, $21, $0d, $20, $01, $e0, $00, $01, $a0, $01,
$20, $81
!by $40, $35, $14, $35, $20, $21, $2d, $12, $01, $e0, $00, $01, $c0, $01,
$20, $61
!by $40, $95, $20, $a1, $e0, $00, $01, $e0, $01, $20, $41, $e0, $20, $21,
$49, $01
!by $e0, $00, $01, $40, $c1, $20, $e1, $a1, $49, $04, $e0, $00, $01, $40,
$01, $a0
!by $01, $20, $e1, $81, $49, $01, $e0, $00, $01, $e0, $40, $01, $20, $e1,
$e1, $20
!by $1b, $a0, $01, $40, $01, $80, $26, $00, $01, $e0, $60, $03, $11, $20,
$01, $e0
!by $00, $01, $40, $01, $a0, $06, $20, $01, $e0, $40, $01, $20, $07, $01,
$16, $e0
!by $e1, $e1, $e1, $c1, $e0, $e0, $e0, $e0, $e0, $20, $1b, $ff
; level 10
!by $62, $e0, $01, $40, $12, $01, $e0, $60, $e2, $42, $1c, $17, $02, $e0,
$01, $40
!by $1b, $01, $e0, $60, $02, $e9, $82, $e0, $01, $20, $19, $11, $01, $e0,
$60, $e2
!by $09, $02, $60, $16, $80, $15, $00, $1b, $01, $00, $41, $e0, $60, $0f,
$14, $15
```

```
!by $14, $15, $14, $15, $14, $15, $02, $e0, $40, $04, $e0, $e0, $20, $e2,
$09, $02
!by $a0, $10, $40, $10, $0d, $10, $e0, $e0, $00, $02, $e9, $02, $60, $15,
$03, $0d
!by $10, $40, $10, $e0, $e0, $20, $e2, $22, $61, $19, $00, $10, $e0, $e0,
$a0, $10
!by $0d, $00, $0d, $00, $0d, $20, $08, $a0, $01, $e0, $e0, $e0, $10, $2d,
$20, $0d
!by $e0, $e0, $e0, $e0, $40, $10, $4d, $20, $2d, $00, $0d, $40, $1b, $01,
$e0, $e0
!by $e0, $00, $10, $0d, $00, $4d, $00, $4d, $32, $19, $11, $01, $e0, $e0,
$e0, $20
!by $10, $2d, $00, $6d, $00, $e2, $62, $e0, $e0, $60, $30, $4d, $00, $2d,
$06, $a0
!by $10, $40, $02, $e0, $e0, $a0, $b0, $15, $00, $10, $00, $10, $00, $10,
$0d, $10
!by $20, $02, $e0, $e0, $e0, $60, $14, $15, $10, $40, $10, $60, $02, $e0,
$e0, $e0
!by $60, $11, $14, $15, $00, $10, $80, $10, $05, $e0, $e0, $e0, $60, $31,
$14, $15
!by $20, $10, $60, $02, $e0, $e0, $e0, $60, $51, $14, $15, $20, $10, $40,
$02, $e0
!by $e0, $e0, $60, $17, $51, $14, $15, $07, $00, $10, $20, $02, $e0, $e0,
$e0, $40
!by $1d
; level 11
!by $e1, $e1, $e1, $e1, $e1, $01, $40, $04, $e0, $40, $0d, $e0, $03, $e0,
$60, $06
!by $21, $00, $18, $21, $20, $31, $0d, $e0, $01, $d1, $80, $01, $60, $01,
$20, $21
!by $20, $01, $00, $01, $71, $c0, $41, $e0, $40, $21, $40, $01, $20, $21,
$20, $01
!by $20, $01, $18, $11, $0d, $11, $c0, $01, $e0, $60, $01, $18, $01, $20,
$01, $20
!by $21, $20, $01, $20, $01, $00, $31, $a0, $01, $00, $01, $00, $01, $e0,
$20, $01
!by $34, $01, $09, $01, $20, $21, $20, $01, $00, $01, $e0, $40, $41, $0d,
$e0, $20
!by $01, $40, $21, $20, $21, $20, $21, $e0, $80, $01, $e0, $60, $01, $20,
$0d, $00
!by $01, $20, $21, $e0, $e0, $00, $01, $e0, $e0, $60, $21, $e0, $e0, $41,
$e0, $e0
!by $40, $21, $e0, $e0, $00, $01, $e0, $e0, $60, $21, $e0, $c0, $01, $00,
$01, $00
!by $01, $60, $35, $e0, $60, $21, $e0, $40, $01, $60, $41, $60, $01, $1a,
$15, $e0
!by $60, $21, $e0, $00, $e1, $e1, $01, $e0, $60, $21, $e0, $40, $01, $60,
$41, $40
!by $18, $01, $e0, $a0, $21, $e0, $c0, $01, $00, $01, $00, $01, $e0, $e0,
$20, $21
!by $a0, $2d, $a0, $14, $00, $1b, $01, $1a, $e0, $e0, $40, $21, $c0, $1c,
```

```
$19, $a0
!by $14, $41, $e0, $e0, $1d, $20, $21, $12, $19, $e0, $c0, $01, $16, $60,
$29, $0d
!by $e0, $20, $19, $07, $e1, $e1, $e1, $e1, $e1, $01
; level 12
!by $e1, $e1, $e1, $e1, $e1, $01, $b1, $e0, $e0, $e0, $c0, $18, $c1, $11,
$40, $0d
!by $e0, $c0, $6d, $a1, $40, $21, $49, $00, $03, $13, $e0, $e0, $00, $0d,
$e1, $61
!by $20, $c1, $91, $e0, $80, $0d, $a1, $18, $20, $41, $20, $21, $b1, $41,
$31, $e0
!by $60, $0d, $21, $27, $80, $41, $20, $e1, $18, $20, $01, $11, $e0, $60,
$0d, $21
!by $12, $20, $0d, $20, $2d, $21, $20, $21, $e0, $01, $31, $e0, $60, $0d,
$a1, $14
!by $81, $20, $21, $18, $1a, $80, $01, $31, $e0, $c0, $81, $14, $61, $00,
$0d, $00
!by $21, $a0, $0d, $01, $31, $e0, $80, $0d, $91, $01, $14, $01, $a0, $21,
$16, $a0
!by $15, $33, $e0, $20, $01, $2d, $b1, $01, $00, $01, $a0, $21, $a0, $0d,
$01, $31
!by $e0, $00, $21, $f1, $01, $00, $01, $a0, $21, $19, $1a, $80, $01, $31,
$e0, $21
!by $f1, $11, $01, $00, $21, $80, $21, $e0, $01, $31, $c0, $01, $f1, $31,
$01, $e0
!by $e1, $19, $20, $01, $11, $c0, $01, $f1, $31, $01, $00, $21, $80, $21,
$b1, $41
!by $31, $c0, $04, $91, $1c, $71, $01, $00, $e1, $a1, $91, $c0, $21, $f1,
$31, $01
!by $e0, $21, $72, $04, $11, $e0, $20, $21, $f1, $51, $01, $a0, $1d, $00,
$c1, $11
!by $e0, $06, $2d, $f1, $71, $01, $00, $0d, $00, $21, $40, $e1, $e1, $e1,
$e1, $e1
!by $01
; level 13
!by $1d, $4d, $00, $0d, $12, $2d, $51, $e0, $40, $11, $e0, $e0, $2d, $20,
$0d, $40
!by $0d, $51, $40, $1b, $07, $1a, $80, $31, $e0, $e0, $4d, $20, $2d, $00,
$51, $40
!by $1b, $07, $1a, $60, $71, $e0, $60, $31, $2d, $00, $4d, $00, $b1, $e0,
$d1, $16
!by $c0, $b1, $01, $00, $01, $f1, $f1, $f1, $11, $01, $00, $01, $f1, $11,
$01, $00
!by $01, $f1, $f1, $f1, $11, $01, $00, $01, $f1, $11, $01, $00, $01, $f1,
$f1, $f1
!by $11, $01, $00, $01, $f1, $21, $00, $21, $20, $01, $f1, $f1, $91, $01,
$00, $01
!by $d1, $00, $01, $20, $21, $20, $81, $18, $f1, $91, $74, $20, $15, $b1,
$00, $01
!by $e0, $00, $06, $01, $40, $f1, $31, $54, $31, $f5, $11, $00, $a1, $00,
$81, $09
```

```
!by $20, $f1, $31, $54, $f1, $11, $15, $11, $e0, $e0, $00, $f1, $74, $b1,
$75, $11
!by $e0, $e0, $00, $f1, $74, $b1, $15, $71, $e0, $e0, $20, $f1, $54, $71,
$55, $71
!by $e0, $e0, $40, $f1, $34, $71, $15, $b1, $e0, $e0, $40, $b1, $74, $71,
$15, $b1
!by $a1, $e0, $80, $31, $40, $94, $71, $15, $b1, $a1, $e0, $60, $03, $11,
$80, $11
!by $00, $b1, $b5, $11, $1c, $40, $21, $60, $4d, $80, $71, $40, $11, $00,
$f1, $51
!by $15, $11, $60, $24, $60, $0d, $12, $0d, $40, $f1, $40, $f5, $75, $11
; level 14
!by $e1, $e1, $e1, $e1, $e1, $01, $08, $18, $20, $02, $20, $33, $18, $f3,
$73, $2d
!by $d3, $c0, $21, $40, $17, $02, $00, $0d, $e0, $e0, $e0, $c0, $21, $82,
$c0, $81
!by $c0, $54, $c0, $41, $09, $21, $e0, $60, $01, $09, $20, $01, $a0, $22,
$03, $22
!by $80, $01, $51, $41, $c0, $19, $21, $e0, $00, $21, $20, $22, $17, $22,
$60, $15
!by $18, $01, $11, $01, $11, $21, $e0, $41, $00, $e1, $01, $00, $19, $82,
$60, $15
!by $00, $09, $31, $08, $21, $80, $10, $e0, $e0, $60, $18, $80, $15, $10,
$11, $18
!by $31, $21, $40, $19, $01, $22, $21, $40, $01, $22, $21, $40, $01, $22,
$21, $40
!by $01, $22, $21, $15, $00, $11, $0d, $00, $21, $e0, $01, $c0, $01, $a0,
$06, $01
!by $40, $18, $09, $00, $18, $01, $15, $60, $21, $e0, $10, $e0, $e0, $e0,
$15, $60
!by $21, $60, $01, $22, $21, $40, $01, $22, $21, $40, $01, $22, $21, $40,
$01, $22
!by $21, $15, $60, $21, $e0, $01, $c0, $01, $80, $0d, $00, $01, $a0, $18,
$01, $15
!by $60, $21, $e0, $10, $c0, $10, $40, $09, $19, $e0, $40, $15, $60, $21,
$60, $01
!by $22, $21, $40, $01, $22, $21, $20, $19, $01, $22, $21, $40, $01, $22,
$21, $15
!by $60, $21, $c0, $09, $01, $40, $1b, $00, $07, $18, $01, $c0, $01, $c0,
$01, $15
!by $60, $21, $1a, $e0, $e0, $e0, $19, $c0, $15, $60, $61, $04, $81, $03,
$81, $04
!by $81, $03, $81, $04, $a1, $00, $24, $61, $16, $40, $06, $01, $20, $07,
$09, $00
!by $01, $06, $60, $01, $60, $09, $01, $60, $09, $01, $42, $60, $32, $e1,
$e1, $e1
!by $c1, $02, $1c, $05, $0d, $1d, $20, $27
; level 15
!by $21, $31, $21, $16, $11, $21, $31, $21, $31, $21, $31, $21, $31, $21,
$31, $14
!by $01, $31, $21, $31, $21, $51, $1d, $18, $14, $18, $01, $11, $01, $00,
```

```
$01, $11
!by $01, $00, $01, $12, $01, $00, $01, $06, $01, $00, $01, $12, $01, $00,
$01, $12
!by $01, $14, $01, $06, $01, $00, $01, $07, $01, $00, $01, $16, $21, $00,
$11, $20
!by $41, $13, $01, $0d, $01, $13, $41, $13, $41, $13, $41, $13, $41, $14,
$41, $13
!by $41, $13, $21, $31, $e0, $80, $04, $e0, $40, $13, $00, $14, $00, $09,
$00, $15
!by $00, $07, $00, $15, $20, $21, $00, $11, $00, $09, $11, $00, $11, $00,
$11, $00
!by $11, $20, $18, $f4, $54, $18, $11, $14, $f1, $11, $00, $31, $00, $12,
$e0, $a0
!by $13, $11, $1a, $11, $09, $11, $1b, $11, $13, $15, $20, $11, $14, $11,
$13, $11
!by $14, $11, $13, $11, $00, $21, $00, $11, $00, $41, $00, $41, $11, $e0,
$11, $e0
!by $f1, $11, $15, $14, $11, $00, $14, $00, $01, $12, $01, $09, $01, $0d,
$01, $40
!by $1a, $e0, $60, $0d, $e0, $00, $11, $00, $21, $00, $11, $01, $d4, $11,
$00, $11
!by $01, $51, $20, $11, $00, $51, $20, $31, $e0, $1b, $13, $31, $00, $14,
$00, $14
!by $80, $14, $51, $19, $51, $13, $31, $00, $11, $0d, $11, $20, $11, $00,
$11, $00
!by $11, $40, $11, $00, $11, $00, $21, $09, $11, $00, $d4, $11, $01, $1b,
$07, $51
!by $20, $13, $00, $51, $e0, $80, $51, $00, $f1, $71, $13, $51, $00, $11,
$60, $11
!by $60, $71, $00, $11, $00, $11, $00, $31, $01, $09, $0d, $00, $0d, $e0,
$00, $15
!by $00, $11, $1b, $11, $1a, $11, $0d, $80, $14, $00, $11, $29, $11, $09,
$11, $00
!by $0d, $00, $21, $11, $20, $0d, $00, $f1, $11, $13, $51, $1a, $11, $40,
$12, $0d
!by $60, $71, $00, $11, $20, $61, $09, $11, $e0, $80, $31, $1b, $71, $40,
$f5, $15
!by $81, $11, $20, $11, $40, $d4, $00, $14, $11, $a0, $03, $11, $00, $15,
$b4, $11
!by $14, $61, $12, $01, $0d, $60, $11, $14, $11, $13, $07, $13, $11, $14,
$00, $11
!by $14, $20, $54, $11, $34, $00, $15, $14, $95, $20, $81, $11, $01, $09,
$21, $20
!by $d4, $00, $51, $04, $14, $11, $14, $11, $14, $20, $15, $14, $1a, $80,
$01, $00
!by $04, $61, $1a, $00, $06, $01, $11, $e0, $60, $15, $54, $11, $01, $00,
$0d, $15
!by $14, $1a, $80, $04, $03, $91, $21, $31, $21, $31, $21, $31, $21, $31,
$21, $31
!by $21, $31, $00, $15, $31, $15, $0d, $15, $00, $04, $03, $04, $03, $04,
$03, $1c
```

```
; check if I/O area is reached
!if * > $cfff {
    !error "Level data goes over $cfff!"
}
```