```
;included from PowerUp.asm

    .include        ".\OSIF.asm"

OSJmpTbl:
    .dw     GetRAM
    .dw     GetKeyPress
    .dw     GetKybdStatus
    .dw     ClearScreen
    .dw     HomeScreen
    .dw     CursorFwd
    .dw     CursorBack
    .dw     EnableCursor
    .dw     DispChar
    .dw     Disp2Hex
    .dw     Disp4Hex
    .dw     NewLine
    .dw     SetVMode
    .dw     ReadSector
    .dw     WriteSector
    .dw     GetDiskInfo
OSJTEnd:

JTENTRIES = (OSJTEnd - OSJmpTbl) / 2

;-----------------------------------------------------------------
;
; Operating System Interface logic
;
; See OSIF.asm for interface details
;
; All registers destroyed
;
;-----------------------------------------------------------------

OSEntryPoint:                           ;save value in A in alternate A'
    ex      af,af'                      ;

.if USE_OSIF_LED_CODES
    ld      a,#LEDOS1
    out     (LEDPORT),a
.endif

    ld      a,c                         ;store call index in A
    exx                                 ;switch to alternate register set
```

```
    ld      c,a                     ;put call index into alternate 'c'

    cp      #JTENTRIES              ;check for valid parameter
    jr      nc,OSBadCall            ;if (call is supported)

    ld      hl,#OSJmpTbl            ;point to jump table
    ld      b,#0x00                 ;
    sla     c                       ;bc = c * 2
    add     hl,bc                   ;hl = pointer to function address
    ld      e,(hl)                  ;
    inc     hl                      ;
    ld      d,(hl)                  ;de = *hl
    ex      de,hl                   ;
    jp      (hl)                    ;jump to function pointer in table
                                    ;  DOES NOT RETURN

OSBadCall:                          ;endif
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS2
    out     (LEDPORT),a
.endif

    ret


GetRAM:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS3
    out     (LEDPORT),a
.endif
    ld      de,(RAMLowAddress)
    ld      hl,(RAMHighAddress)
    ret

GetKeyPress:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS4
    out     (LEDPORT),a
.endif
    call    ReadKybd
    ret

GetKybdStatus:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS5
```

```
    out     (LEDPORT),a
.endif
    ret

ClearScreen:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS6
    out     (LEDPORT),a
.endif
    call    ClrLineBuffer
    ret

HomeScreen:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS7
    out     (LEDPORT),a
.endif
    ld      a,#DISPHOME             ;
    call    DisplayA                ;move the cursor to the home posit
    ret

CursorFwd:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS8
    out     (LEDPORT),a
.endif
    ex      af,af'                  ;restore the passed value in A
    ld      b,a
CFLoop:                             ;repeat
    call    DispFS                  ;  move cursor forward
    djnz    CFLoop                  ;until (count satisfied)
    ret

CursorBack:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOS9
    out     (LEDPORT),a
.endif
    ex      af,af'                  ;restore the passed value in A
    ld      b,a
CBLoop:                             ;repeat
    call    DispBS                  ;  move cursor backward
    djnz    CBLoop                  ;until (count satified)
    ret
```

```
EnableCursor:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOSa
    out     (LEDPORT),a
.endif
    ex      af,af'                      ;restore the passed value in A
    ld      (FlashCursor),a
    ret

DispChar:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOSb
    out     (LEDPORT),a
.endif
    ex      af,af'                      ;restore the passed value in A
    call    CmdWRChar
    ret

Disp2Hex:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOSc
    out     (LEDPORT),a
.endif
    ex      af,af'                      ;restore the passed value in A
    call    HexDispA
    ret

Disp4Hex:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOSd
    out     (LEDPORT),a
.endif
    exx                                 ;swap register banks to recover pa
    call    DisplayDE
    ret

NewLine:
.if USE_OSIF_LED_CODES
    ld      a,#LEDOSe
    out     (LEDPORT),a
.endif
    call    CmdNewLine
    ret

SetVMode:
```

```
 .if USE_OSIF_LED_CODES
    ld      a,#LEDOSf
    out     (LEDPORT),a
 .endif
    ret

 .if 0
 ; TODO ADD RANGE CHECKING ON SECTOR NUMBERS IN READ/WRITE SECTOR BELOW
 ; NOT ADDED NOW BECAUSE IT MAKES CODE EXCEED THE ROM SIZE
 ; SHOULD BE SOMETHING LIKE THIS
    push    hl                          ;save hl
    scf                                 ;set C flag
    ld      hl,(HDBASect)               ;get sector number just above last
    sbc     hl,de                       ;set C flag if DE >= HL (i.e. an i
    pop     hl                          ;restore hl
    jr      c,RSBSErr                   ;if (sector is valid for the OS)
 .endif


 ; input A = drive (0-A:, 1-B:, 2-C:, ...)
 ;       HL = pointer to read buffer
 ;       DE = D0:D16 of sector number
 ; output A = 0 on success, !0 = error code
 ReadSector:
 .if SUPPORT_HARD_DISK
    call    HDReadSector                ;  read a sector of data from the
    xor     a
 .else
    ld      a,#0x01
 .endif
    ret

 ; input A = drive (0-A:, 1-B:, 2-C:, ...)
 ;       HL = pointer to write buffer
 ;       DE = D0:D16 of sector number
 ; output A = 0 on success, !0 = error code
 WriteSector:
 .if SUPPORT_HARD_DISK
    call    HDWriteSector               ;write a sector of data to the har
    xor     a
 .else
    ld      a,#0x01
 .endif
    ret
```

```
 ; input A = drive (0-A:, 1-B:, 2-C:, ...)
 ; output A = 0 on success, !0 = error code
 ;        BC = last usable sector number
 ;        DE = #of bytes per sector
 GetDiskInfo:
 .if SUPPORT_HARD_DISK
     call    HDGetInfo                   ;get the disk info
     xor     a
 .else
     ld      a,#0x01
 .endif
     ret
```