

Anlage A:

CP/M 2.2c CBIOS

```

*****
;* CBIOS * CPMSYS1 * Thomas Holte * 831126 *
*****
;*
;* CUSTOMIZED BIOS FOR CP/M 2.2 ON THE *
;* ===== *
;*
;* GENIE III MICROCOMPUTER SYSTEM *
;* ===== *
;*
;*
;* Thomas Holte (C) SoftCream GmbH Version 1.2 *
;*
*****

```

.Z80

0000'

ASEG  
ORG 0100H

.PHASE 0F400H ;start of BIOS

;BIOS jump vector:

F400	C3 F7E7	BOOT: JP \$BOOT	;arrive here from cold start load
F403	C3 F433	WBOOT: JP \$WBOOT	;arrive here for warm start
F406	C3 F4B4	CONST: JP \$CONST	;check for console char ready
F409	C3 F4D8	CONIN: JP \$CONIN	;read console character in
F40C	C3 F4F6	CONOUT: JP \$CONOUT	;write console character out
F40F	C3 F514	LIST: JP \$LIST	;write listing character out
F412	C3 F530	PUNCH: JP \$PUNCH	;write character to punch device
F415	C3 F545	READER: JP \$READER	;read reader device
F418	C3 F560	HOME: JP \$HOME	;move to track 0 on selected disk
F41B	C3 F575	SELDSK: JP \$SELDSK	;select disk drive
F41E	C3 F596	SETTRK: JP \$SETTRK	;set track number
F421	C3 F59B	SETSEC: JP \$SETSEC	;set sector number
F424	C3 F5A0	SETDMA: JP \$SETDMA	;set DMA address
F427	C3 F5A5	READ: JP \$\$READ	;read selected sector
F42A	C3 F5BC	WRITE: JP \$\$WRITE	;write selected sector
F42D	C3 F6EF	LISTST: JP \$LISTST	;return list status
F430	C3 F720	SECTTRAN: JP \$SECTTRAN	;sector translate subroutine

;ASCII control codes:

0000	NUL	EQU 00H	;null
0003	ETX	EQU 03H	;end of text
0007	BEL	EQU 07H	;bell
0008	BS	EQU 08H	;backspace
0009	HT	EQU 09H	;horizontal tabulation
000A	LF	EQU 0AH	;line feed
000B	VT	EQU 0BH	;vertical tabulation
000C	FF	EQU 0CH	;form feed
000D	CR	EQU 0DH	;carriage return
0018	CAN	EQU 18H	;cancel
0019	EM	EQU 19H	;end of medium
001A	SUB	EQU 1AH	;substitute
001B	ESC	EQU 1BH	;escape
001E	RS	EQU 1EH	;record separator
007F	DEL	EQU 7FH	;delete

```

F433          $WBOOT:
              ;=====

              ;arrive here for warm start:
F433 31 DDFE          LD SP,CCP-2          ;set temporary stack pointer
F436 AF              XOR A                ;clear accu
F437 0E 0C           LD C,12              ;vector # --> reg. C
F439 21 DE00         LD HL,BOOT-1600H     ;CCP-entry-point --> reg. HL
F43C 11 0200         LD DE,512           ;sector length --> reg. DE
F43F 32 FFDA         LD (DRIVE),A         ; drive # = 0 (CP/M-drive A:)
F442 32 FFDB         LD (TRACK),A        ; track # = 0 (first double density track)
F445 3C              INC A
F446 32 FFDD         LDSYS: LD (SECTOR),A ;sector # = 1
F449 22 FFDB         LD (BUFFER),HL      ;store buffer address
F44C F5              PUSH AF              ;save accu
F44D CD FE96         CALL $ROMIO          ;read selected sector into main memory
F450 B7              OR A                 ;disk error ?
F451 20 3F           JR NZ,LDERR
F453 F1              POP AF              ;restore accu
F454 3C              INC A                ;increment sector #
F455 19              ADD HL,DE           ;bump buffer ptr
F456 FE 0C           CP 12                ;all 11 system sectors read ?
F458 20 EC           JR NZ,LDSYS         ;read next, if not
F45A 21 F7E7         LD HL,HSTBUF       ;restore disk buffer address
F45D 22 FFD8         LD (BUFFER),HL

              ;part of sector deblocking algorithm:
F460 AF              XOR A                ;0 to accumulator
F461 32 F6E2         LD (HSTACK),A        ;host buffer inactive
F464 32 F6E4         LD (UNACNT),A       ;clear unalloc count

              ;continue warm start:
F467 2A 0003         LD HL,(0003H)       ;IOBYTE, current user no. & drive --> HL
F46A 22 F48D         LD ($START+3),HL    ;store it into parameter block

              ;initialize interrupt service routine:
F46D F3              CBOOTIN:DI          ;disable interrupts
F46E ED 56           IM 1                 ;non maskable interrupt (RST 38)
F470 21 0038         LD HL,0038H        ;interrupt vector --> reg. HL
F473 36 F3           LD (HL),DI         ;0038 DI
F475 23              INC HL
F476 36 C9           LD (HL),RET         ;0039 RET

              ;initialize addresses 0000 - 0007:
F478 21 F48A         LD HL,$START       ;first seven bytes --> reg. HL
F47B 11 0000         LD DE,0000H        ;address 0000 --> reg. DE
F47E 01 0008         LD BC,8            ;byte count --> reg. BC
F481 ED B0           LDIR                ;transfer first seven bytes

              ;exit to CCP:
F483 3A 0004         LD A,(0004H)       ;current drive --> accu
F486 4F              LD C,A              ;current drive --> reg. C
F487 C3 DE00         JP CCP

```

```

F48A C3 F403      $START: JP  WBOOT      ;warm start
F48D 81          DEFB 1000001B ;IOBYTE
                                ;CON: = CRT:   (console)
                                ;RDR: = TTY:   (RS232C )
                                ;PLN: = TTY:
                                ;LST: = LPT:   (printer)
F48E 00          DEFB 00H      ;bits 7-4: current user no.
                                ;bits 3-0: current drive
F48F C3 E606      JP  BDOS      ;primary entry point to CP/M for transient
                                ;programs

                                ;disk error during WARMBOOT:
F492 0E 1A      LDERR: LD  C,SUB
F494 CD F4F6      CALL $CONOUT      ;clear screen
F497 3E F0      LD  A,OF0H
F499 F3          DI          ;disable interrupts
F49A D3 FA      OUT ($BSEL),A ;turn on video RAM
F49C 21 F4AA      LD  HL,MSG      ;^message          --> reg. HL
F49F 11 3C00      LD  DE,3C00H      ;^first screen position --> reg. DE
F4A2 01 000A      LD  BC,$CONST-MSG ;length of message   --> reg. BC
F4A5 ED B0      LDIR      ;output error message
F4A7 C3 0083      JP  0083H      ;secret ROM call (alarm)
F4AA 44 49 53 4B MSG:  DEFM 'DISK ERROR'
F4AE 20 45 52 52
F4B2 4F 52

```

```

F4B4          $CONST:
              ;=====

              ;check for console char ready:
F4B4  11 F4BD          LD  DE,CONTB1          ;jump vector --> reg. DE
F4B7  3A 0003         LD  A,(IOBYTE)         ;IOBYTE  --> accu
F4BA  C3 F714         JP  JUMPIO
F4BD  F4C5 F4C9       CONTB1: DEFW TTYSTI,CRTSTI,BATST,UC1ST
F4C1  F4D3 F4D0

F4C5  0E 08          TTYSTI: LD  C,8              ;vector # --> reg. C
F4C7  18 02          JR  CRTSTI+2            ;test RS232C status (input)
F4C9  0E 03          CRTSTI: LD  C,3           ;vector # --> reg. C
F4CB  CD FE96        CALL $ROMIO           ;scanner routine
F4CE  B7             OR  A              ;key pressed ?
F4CF  C8             RET  Z              ;return if no key pressed
F4D0  3E FF          UC1ST: LD  A,OFFH       ;FF = char pending
F4D2  C9             RET                  ;return to caller
F4D3  11 F550        BATST: LD  DE,RDRTB1      ;jump vector --> reg. DE
F4D6  18 70          JR  $READER+3

F4D8          $CONIN:
              ;=====

              ;read console character in:
F4D8  11 F4E1          LD  DE,CONTB2          ;jump vector --> reg. DE
F4DB  3A 0003         LD  A,(IOBYTE)         ;IOBYTE  --> accu
F4DE  C3 F714         JP  JUMPIO
F4E1  F4E9 F4EE       CONTB2: DEFW TTYIN,CRTIN,$READER,UC1IN
F4E5  F545 F4F3

F4E9  0E 09          TTYIN: LD  C,9              ;vector # --> reg. C
F4EB  C3 FE96        JP  $ROMIO           ;read char from RS232C and return to caller
F4EE  0E 04          CRTIN: LD  C,4           ;vector # --> reg. C
F4F0  C3 FE96        JP  $ROMIO           ;read char and return to caller
F4F3  3E 1A          UC1IN: LD  A,SUB         ;EOF --> accu
F4F5  C9             RET                  ;return to caller

F4F6          $CONOUT:
              ;=====

              ;write console character out:
F4F6  11 F4FF          LD  DE,CONTB3          ;jump vector --> reg. DE
F4F9  3A 0003         LD  A,(IOBYTE)         ;IOBYTE  --> accu
F4FC  C3 F714         JP  JUMPIO
F4FF  F507 F50D       CONTB3: DEFW TTYOUT,CRTOUT,$LIST,UC1OUT
F503  F514 F513

F507  79             TTYOUT: LD  A,C           ;character --> accu
F508  0E 0B          LD  C,11             ;vector # --> reg. C
F50A  C3 FE96        JP  $ROMIO           ;write char to RS232C and return to caller
F50D  79             CRTOUT: LD  A,C          ;character --> accu
F50E  0E 05          LD  C,5              ;vector # --> reg. C
F510  C3 FE96        JP  $ROMIO           ;output char and return to caller
F513  C9             UC1OUT: RET          ;return to caller

```

```

F514          $LIST:
              ;=====

              ;write listing character out:
F514  11 F522          LD  DE,LSTTB1          ;jump vector --> reg. DE
F517  3A 0003          LD  A,(IOBYTE)         ;IOBYTE    --> accu
F51A  06 06           LD  B,6
F51C  1F              RRA
F51D  10 FD           DJNZ $-1
F51F  C3 F714         JP  JUMPIO
F522  F507 F50D       LSTTB1: DEFW TTYOUT,CRTOUT,LPTOUT,UL1OUT
F526  F52A F513

F52A  79              LPTOUT: LD  A,C          ;character --> accu
F52B  0E 07           LD  C,7          ;vector # --> reg. C
F52D  C3 FE96         JP  $ROMIO         ;print char and return to caller
F513          UL1OUT EQU  UC1OUT

F530          $PUNCH:
              ;=====

              ;write character to punch device:
F530  11 F53D          LD  DE,PUNTAB          ;jump vector --> reg. DE
F533  3A 0003          LD  A,(IOBYTE)         ;IOBYTE    --> accu
F536  1F              RRA
F537  1F              RRA
F538  1F              RRA
F539  1F              RRA
F53A  C3 F714         JP  JUMPIO
F53D  F507 F513       PUNTAB: DEFW TTYOUT,PTPOUT,UR1OUT,UR2OUT
F541  F513 F513

F513          PTPOUT EQU  UL1OUT
F513          UR1OUT EQU  UL1OUT
F513          UR2OUT EQU  UL1OUT

F545          $READER:
              ;=====

              ;read reader device:
F545  11 F558          LD  DE,RDRTB2          ;jump vector --> reg. DE
F548  3A 0003          LD  A,(IOBYTE)         ;IOBYTE    --> accu
F54B  1F              RRA
F54C  1F              RRA
F54D  C3 F714         JP  JUMPIO
F550  F4C5 F4D0       RDRTB1: DEFW TTYST1,PTRST,UR1ST,UR2ST
F554  F4D0 F4D0
F558  F4E9 F4F3       RDRTB2: DEFW TTYIN,PTRIN,UR1IN,UR2IN
F55C  F4F3 F4F3

F4D0          PTRST EQU  UC1ST
F4D0          UR1ST EQU  UC1ST
F4D0          UR2ST EQU  UC1ST

F4F3          PTRIN EQU  UC1IN
F4F3          UR1IN EQU  UC1IN
F4F3          UR2IN EQU  UC1IN

```

```

;|=====|
;| sector deblocking algorithms for CP/M 2.0 !
;| (adapted from CP/M-Manual, Appendix 6) !
;|=====|

FFDA      HSTDSK EQU DRIVE
FFDB      HSTTRK EQU TRACK
FFDD      HSTSEC EQU SECTOR

F560      $HOME:
;=====

;move to track zero on selected disk:
F560      3A F6DD      LD A,(SEKDSK)      ;selected drive # --> accu
F563      0E 10       LD C,16          ;vector # --> reg. C
F565      CD FE96     CALL $ROMIO      ;secret ROM call to calculate ^DCT
F568      DD CB 03 86 RES D,(IX+3)     ;reset selected drive
F56C      3A F6E3     LD A,(HSTWRT)     ;check for pending write
F56F      B7         OR A
F570      CD         RET NZ
F571      32 F6E2     LD (HSTACT),A     ;clear host active flag
F574      C9         RET

F575      $SELDSK:
;=====

;select disk drive:
F575      21 0000     LD HL,0000H      ;initialize ^DPBASE
F578      3A F0FF     LD A,(MAXDRV)     ;maximum drive # --> accu
F57B      B9         CP C          ;legal drive selected ?
F57C      D8         RET C        ;return, if not
F57D      79         LD A,C        ;selected disk number
F57E      32 F6DD     LD (SEKDSK),A     ;seek disk number
F581      6F         LD L,A        ;disk number to HL
F582      29         ADD HL,HL     ;multiply by 4
F583      29         ADD HL,HL
F584      E5         PUSH HL      ;save it
F585      11 F0EF     LD DE,CPMHST  ;base of table of CP/M to host disk constants
F588      19         ADD HL,DE     ;HL = CPMHST(CURDSK)
F589      22 F594     LD (DSKVEC),HL ;store it
F58C      E1         POP HL       ;restore disk number * 8
F58D      29         ADD HL,HL     ;multiply by 4
F58E      29         ADD HL,HL
F58F      11 FE00     LD DE,DPBASE  ;base of parm block
F592      19         ADD HL,DE     ;HL = DPB(CURDSK)
F593      C9         RET
F594      0000      DSKVEC: DEFW D ;temp mem for ^table of CP/M to host disk
;constants

F596      $SETTRK:
;=====
;set track number:
F596      ED 43 F6DE LD (SEKTRK),BC ;track to seek
F59A      C9         RET

```

```

F598                                $SETSEC:
;=====

;set sector number:
F598 79                             LD  A,C
F59C 32 F6E0                         LD  (SEKSEC),A      ;sector to seek
F59F C9                             RET

F5A0                                $SETDMA:
;=====

;set DMA address:
F5A0 ED 43 F6ED                       LD  (DMAADR),BC
F5A4 C9                             RET

F5A5                                $$READ:
;=====

;get DSKHST pointer:
F5A5 FD 2A F594                       LD  IY,(DSKVEC)

;read selected sector:
F5A9 AF                             XOR  A
F5AA 32 F6E4                         LD  (UNACNT),A
F5AD 3E 01                             LD  A,1
F5AF 32 F6EB                         LD  (READOP),A      ;read operation
F5B2 32 F6EA                         LD  (RSFLAG),A      ;must read data
F5B5 3E 02                             LD  A,WRUAL
F5B7 32 F6EC                         LD  (WRTYPE),A      ;treat as unalloc
F5BA 18 6A                             JR  RWOPER           ;to perform the read

F5BC                                $$WRITE:
;=====

;BDOS constants on entry to write
0000 WRALL EQU 0                       ;write to allocated
0001 WRDIR EQU 1                       ;write to directory
0002 WRUAL EQU 2                       ;write to unallocated

;get DSKHST pointer:
F5BC FD 2A F594                       LD  IY,(DSKVEC)

;write selected sector:
F5C0 AF                             XOR  A              ;0 to accumulator
F5C1 32 F6EB                         LD  (READOP),A      ;not a read operation
F5C4 79                             LD  A,C              ;write type in C
F5C5 32 F6EC                         LD  (WRTYPE),A
F5C8 FE 02                             CP  WRUAL            ;write unallocated ?
F5CA 2D 18                             JR  NZ,CHKUNA        ;check for unalloc

```



```

;write to unallocated, set parameters
F5CC  FD 7E 0D          LD  A,(IY)           ;next unalloc recs
F5CF  32 F6E4          LD  (UNACNT),A
F5D2  3A F6DD          LD  A,(SEKDSK)      ;disk to seek
F5D5  32 F6E5          LD  (UNADSK),A      ;UNADSK = DRIVE
F5D8  2A F6DE          LD  HL,(SEKTRK)
F5DB  22 F6E6          LD  (UNATRK),HL     ;UNATRK = TRACK
F5DE  3A F6E0          LD  A,(SEKSEC)
F5E1  32 F6E8          LD  (UNASEC),A      ;UNASEC = SEKSEC

;check for write to unallocated sector:
F5E4  3A F6E4          CHKUNA: LD A,(UNACNT) ;any unalloc remain ?
F5E7  B7              OR  A
F5E8  28 34           JR  Z,ALLOC         ;skip if not

;more unallocated records remain:
F5EA  3D             DEC  A             ;UNACNT = UNACNT - 1
F5EB  32 F6E4          LD  (UNACNT),A
F5EE  3A F6DD          LD  A,(SEKDSK)      ;same disk ?
F5F1  21 F6E5          LD  HL,UNADSK
F5F4  BE             CP  (HL)           ;SEKDSK = UNADSK ?
F5F5  20 27           JR  NZ,ALLOC       ;skip if not

;disks are the same:
F5F7  21 F6E6          LD  HL,UNATRK
F5FA  CD F6BF          CALL SKTRKCMP       ;HSTRK = UNATRK ?
F5FD  20 1F           JR  NZ,ALLOC       ;skip if not

;tracks are the same:
F5FF  3A F6E0          LD  A,(SEKSEC)      ;same sector ?
F602  21 F6E8          LD  HL,UNASEC
F605  BE             CP  (HL)           ;SEKSEC = UNASEC ?
F606  20 16           JR  NZ,ALLOC       ;skip if not

;match, move to next sector for future ref:
F608  34             INC  (HL)          ;UNASEC = UNASEC + 1
F609  7E             LD  A,(HL)         ;end of track ?
F60A  FD BE D1        CP  (IY+1)         ;count CP/M sectors
F60D  38 09           JR  C,NOOVF        ;skip if no overflow

;overflow to next track:
F60F  36 0D           LD  (HL),0         ;UNASEC = 0
F611  2A F6E6          LD  HL,(UNATRK)
F614  23             INC  HL
F615  22 F6E6          LD  (UNATRK),HL    ;UNATRK = UNATRK + 1

;match found, mark as unnecessary read:
F618  AF             NOOVF: XOR A        ;0 to accumulator
F619  32 F6EA          LD  (RSFLAG),A     ;RSFLAG = 0
F61C  16 DB           JR  RWOPER        ;to perform the write

;not an unallocated record, requires pre-read:
F61E  AF             ALLOC: XOR A       ;0 to accum
F61F  32 F6E4          LD  (UNACNT),A     ;UNACNT = 0
F622  3C             INC  A             ;1 to accum
F623  32 F6EA          LD  (RSFLAG),A     ;RSFLAG = 1

```

```

;common code for read and write follows:
;enter here to perform the READ/WRITE:
F626 AF RWOPER: XOR A ;zero to accum
F627 32 F6E9 LD (ERFLAG),A ;no errors (yet)
F62A 3A F6ED LD A,(SEKSEC) ;compute host sector
F62D FD 46 03 LD B,(IY+3)
F630 04 INC B
F631 05 DEC B
F632 28 04 JR Z,$+6
F634 CB 3F SRL A ;shift right
F636 10 FC DJNZ $-2
F638 32 F6E1 LD (SEKHST),A ;host sector to seek

;active host sector ?
F63B 21 F6E2 LD HL,HSTACT ;host active flag
F63E 7E LD A,(HL)
F63F 36 01 LD (HL),1 ;always becomes 1
F641 B7 OR A ;was it already ?
F642 28 21 JR Z,FILHST ;fill host if not

;host buffer active, same as seek buffer ?
F644 3A F6DD LD A,(SEKDSK)
F647 21 FFDA LD HL,HSTDSK ;same disk ?
F64A BE CP (HL) ;SEKDSK = HSTDSK ?
F64B 20 11 JR NZ,NOMATCH

;same disk, same track ?
F64D 21 FFDB LD HL,HSTTRK ;same track ?
F650 CD F6BF CALL SKTRKCMP ;SEKTRK = HSTTRK ?
F653 20 09 JR NZ,NOMATCH

;same disk, same track, same buffer ?
F655 3A F6E1 LD A,(SEKHST)
F658 21 FFDD LD HL,HSTSEC ;same buffer ?
F65B BE CP (HL) ;SEKHST = HSTSEC ?
F65C 28 24 JR Z,MATCH

;proper disk, but not correct sector:
F65E 3A F6E3 NOMATCH:LD A,(HSTWRT) ;host written ?
F661 B7 OR A
F662 C4 F6CB CALL NZ,WRITEHST ;clear host buff

;may have to fill the host buffer
F665 3A F6DD FILHST: LD A,(SEKDSK)
F668 32 FFDA LD (HSTDSK),A
F66B 2A F6DE LD HL,(SEKTRK)
F66E 22 FFDB LD (HSTTRK),HL
F671 3A F6E1 LD A,(SEKHST)
F674 32 FFDD LD (HSTSEC),A
F677 3A F6EA LD A,(RSFLAG) ;need to read ?
F67A B7 OR A
F67B C4 F6D4 CALL NZ,READHST ;yes, if 1
F67E AF XOR A ;0 to accum
F67F 32 F6E3 LD (HSTWRT),A ;no pending write

```

```

;copy data to or from buffer:
F682 3A F6E0 MATCH: LD A,(SEKSEC) ;mask buffer number
F685 FD A6 02 AND (IY+2) ;least signif bits
F688 6F LD L,A ;ready to shift
F689 26 00 LD H,0 ;double count
F68B 06 07 LD B,7 ;shift left 7
F68D 29 ADD HL,HL
F68E 10 FD DJNZ $-1

;HL has relative host buffer address:
F690 11 F7E7 LD DE,HSTBUF
F693 19 ADD HL,DE ;HL = host address
F694 ED 5B F6ED LD DE,(DMAADR) ;get/put CP/M data
F698 01 0080 LD BC,128 ;length of move
F69B 3A F6EB LD A,(READOP) ;which way ?
F69E B7 OR A
F69F 20 06 JR NZ,RWMOVE ;skip if read

;write operation, mark and switch direction:
F6A1 3E 01 LD A,1
F6A3 32 F6E3 LD (HSTWRT),A ;HSTWRT = 1
F6A6 EB EX DE,HL ;source/dest swap

;BC initially 128, DE is dest, HL is source:
F6A7 ED B0 RWMOVE: LDIR

;data has been moved to/from host buffer:
F6A9 3A F6EC LD A,(WRTYPE) ;write type
F6AC FE 01 CP WRDIR ;to directory ?
F6AE 3A F6E9 LD A,(ERFLAG) ;in case of errors
F6B1 C0 RET NZ ;no further processing

;clear host buffer for directory write:
F6B2 B7 OR A ;errors ?
F6B3 C0 RET NZ ;skip if so
F6B4 AF XOR A ;0 to accum
F6B5 32 F6E3 LD (HSTWRT),A ;buffer written
F6B8 CD F6CB CALL WRITEHST
F6BB 3A F6E9 LD A,(ERFLAG)
F6BE C9 RET

```

```

;utility subroutine for 16-bit compare:

;HL = UNATRK or HSTTRK, compare with SEKTRK
SKTRKCMP:
F6BF      EB          EX  DE,HL
F6C0      21 F6DE     LD  HL,SEKTRK
F6C3      1A          LD  A,(DE)          ;low byte compare
F6C4      BE          CP  (HL)          ;same ?
F6C5      C0          RET  NZ          ;return if not

;low bytes equal, test high 1s:
F6C6      13          INC  DE
F6C7      23          INC  HL
F6C8      1A          LD  A,(DE)
F6C9      BE          CP  (HL)          ;sets flags
F6CA      C9          RET

;WRITEHST performs the physical write to the host disk, READHST reads the
;physical disk:

;HSTDSK = host disk #, HSTTRK = host track #
;HSTSEC = host sect #. Write "HSTSIZ" bytes
;from HSTBUF and return error flag in ERFLAG.
;Return ERFLAG non-zero if error.
WRITEHST:
F6CB      0E 0D          LD  C,13          ;vector # --> reg. C
F6CC      CD FE96       CALL $ROMIO      ;write sector
F6CD      32 F6E9       LD  (ERFLAG),A   ;return error flag
F6D3      C9          RET

;HSTDSK = host disk #, HSTTRK = host track #
;HSTSEC = host sect #. Read "HSTSIZ" bytes
;into HSTBUF and return error flag in ERFLAG.
READHST:
F6D4      0E 0C          LD  C,12          ;vector # --> reg. C
F6D6      CD FE96       CALL $ROMIO      ;read sector
F6D9      32 F6E9       LD  (ERFLAG),A   ;return error flag
F6DC      C9          RET

F6DD      00          SEKDSK: DEFB 0          ;seek disk number
F6DE      0000        SEKTRK: DEFW 0          ;seek track number
F6E0      00          SEKSEC: DEFB 0          ;seek sector number

F6E1      00          SEKSHST: DEFB 0         ;seek shr secshf
F6E2      00          HSTACT: DEFB 0         ;host active flag
F6E3      00          HSTWRT: DEFB 0         ;host written flag

F6E4      00          UNACNT: DEFB 0         ;unalloc rec cnt
F6E5      00          UNADSK: DEFB 0         ;last unalloc disk
F6E6      0000        UNATRK: DEFW 0         ;last unalloc track
F6E8      00          UNASEC: DEFB 0         ;last unalloc sector

F6E9      00          ERFLAG: DEFB 0         ;error reporting
F6EA      00          RSFLAG: DEFB 0         ;read sector flag
F6EB      00          READOP: DEFB 0         ;! if READ operation
F6EC      00          WRTYPE: DEFB 0         ;WRITE operation type
F6ED      0080        DMAADR: DEFW 0080H     ;last dma address

```

```

F6EF          $LISTST:
              ;=====

              ;return list status:
F6EF  11 F6FC          LD  DE,LSTTB2          ;jump vector --> reg. DE
F6F2  3A 0003          LD  A,(IOBYTE)        ;IOBYTE --> accu
F6F5  06 06           LD  B,6
F6F7  1F              RRA
F6F8  10 FD           DJNZ $-1
F6FA  18 18           JR   JUMPIO
F6FC  F704 F4D0        LSTTB2: DEFW TTYSTO,CRTSTO,LPTST,UL1ST
F700  F708 F710

F704  0E 0A           TTYSTO: LD  C,10          ;vector # --> reg. C
F706  18 02           JR   LPTST+2
F4D0                      CRTSTO EQU UC1ST
F708  0E 06           LPTST: LD  C,6          ;vector # --> reg. C
F70A  CD FE96         CALL $ROMIO          ;test list status
F70D  B7              OR   A
F70E  28 02           JR   Z,LPTST1
F710  AF              XOR  A          ;list device not ready
F711  C9              RET
F712  2F              LPTST1: CPL          ;list device ready
F713  C9              RET
F710          UL1ST EQU LPTST1-2

F714  E6 03           JUMPIO: AND 3          ;mask IOBYTE
F716  26 00           LD  H,0          ;IOBYTE --> reg. HL
F718  6F              LD  L,A
F719  29              ADD  HL,HL          ;IOBYTE * 2
F71A  19              ADD  HL,DE          ;add table vector
F71B  7E              LD  A,(HL)
F71C  23              INC  HL
F71D  66              LD  H,(HL)
F71E  6F              LD  L,A
F71F  E9              JP   (HL)          ;perform I/O routine

F720          $SECTTRAN:
              ;=====

              ;sector translate subroutine:
F720  60              LD  H,B          ;logical sector # --> reg. HL
F721  69              LD  L,C
F722  7A              LD  A,D          ;exists translate table ?
F723  B3              OR   E
F724  C8              RET  Z          ;return if no translation
F725  19              ADD  HL,DE          ;compute table entry
F726  6E              LD  L,(HL)          ;physical sector # --> reg. HL
F727  26 00          LD  H,0
F729  C9              RET

              .DEPHASE

              ORG  04E7H

              .PHASE 0F7E7H

```

```

;disk buffer:
F7E7 HSTBUF:

F7E7 $BOOT:
;=====

;arrive here from cold start load:
.FLIST
F8D9 C3 F46D JP CB00TIN ;execute 2nd part of cold start loader
.FLIST
FA70 DEFS 1024-(ENDMSG-HSTBUF)

;dir buf:
FBE7 DIRBUF: DEFS 128

;check vectors:
FC67 CSV0: DEFS 48
FC97 CSV1: DEFS 48
FCC7 CSV2: DEFS 48
FCF7 CSV3: DEFS 48

;alloc vectors:
FD27 ALV0: DEFS 50
FD59 ALV1: DEFS 50
FD8B ALV2: DEFS 50
FDBD ALV3: DEFS 50
```

```

; utility macro to compute sector mask
SMASK MACRO HBLK
; compute log2(HBLK), return @X as result
; (2 ** @X = HBLK on return)
@Y DEFL HBLK
@X DEFL 0
; count right shifts of @Y until = 1
REPT 8
COND @Y EQ 1
EXITM
ENDC
; @Y is not 1, shift right one position
@Y DEFL @Y SHR 1
@X DEFL @X+1
ENDM
ENDM

;CP/M to host disk constants for drive 0:
FDEF 10 CPMHST: DEFB 16 ;CP/M sectors/block
FDF0 50 DEFB 80 ;CP/M sectors/track
FDF1 03 DEFB 3 ;sector mask
SMASK 4 ;compute sector mask
FDF2 02 DEFB @X ;log2(CP/M sectors/host buff)

;CP/M to host disk constants for drive 1:
FDF3 10 50 03 DEFB 16,80,3
SMASK 4
FDF6 02 DEFB @X

;CP/M to host disk constants for drive 2:
FDF7 08 1A 00 DEFB 8,26,0
SMASK 1
FDF8 00 DEFB @X

;CP/M to host disk constants for drive 3:
FDFB DEFS 4

```

```
        ;disk parameter blocks:

FDFF    02                MAXDRV: DEFB 2                ;number of disk drives

        ;disk parameter header for drive 0:
FE00    0000              DPBASE: DEFW 0000H            ;translate table
FE02    0000 0000                DEFW 0,0,0            ;scratch area
FE06    0000
FE08    FBE7                DEFW DIRBUF              ;dir buff
FE0A    FE40                DEFW DPB0                ;parm block
FE0C    FC67                DEFW CSV0                ;check vector
FE0E    FD27                DEFW ALV0                ;alloc vector

        ;disk parameter header for drive 1:
FE10    0000 0000              DEFW 0,0,0,0,DIRBUF,DPB1,CSV1,ALV1
FE14    0000 0000
FE18    FBE7 FE4F
FE1C    FC97 FD59

        ;disk parameter header for drive 2:
FE20    FE7C 0000              DEFW XLT,0,0,0,DIRBUF,DPB2,CSV2,ALV2
FE24    0000 0000
FE28    FBE7 FE5E
FE2C    FCC7 FD8B

        ;disk parameter header for drive 3:
FE30    0000 0000              DEFW 0,0,0,0,DIRBUF,DPB3,CSV3,ALV3
FE34    0000 0000
FE38    FBE7 FE6D
FE3C    FCF7 FDBD
```



```

;disk parameter block for drive 0:
FE40 0050      DPB0:  DEFW 80          ;sec per track
FE42 04        DEFB 4           ;block shift
FE43 0F        DEFB 15          ;block mask
FE44 00        DEFB 0           ;extnt mask
FE45 018A     DEFW 394          ;disk size - 1
FE47 00BF     DEFW 191          ;directory max
FE49 E0       DEFB 0E0H        ;alloc 0
FE4A 00       DEFB 0           ;alloc 1
FE48 0030     DEFW 48           ;check size
FE4D 0001     DEFW 1           ;offset

;disk parameter block for drive 1:
FE4F 0050      DPB1:  DEFW 80          ;sec per track
FE51 04 0F 00 DEFB 4,15,0       ;block shift
FE54 018A 00BF DEFW 394,191     ;disk size - 1
FE58 E0 00     DEFB 0E0H,0     ;alloc 0
FE5A 0030 0001 DEFW 48,1       ;check size

;disk parameter block for drive 2:
FE5E 001A     DPB2:  DEFW 26          ;sec per track
FE60 03 07 00 DEFB 3,7,0       ;block shift
FE63 00F2 003F DEFW 242,63     ;disk size - 1
FE67 C0 00     DEFB 0C0H,0     ;alloc 0
FE69 0010 0002 DEFW 16,2       ;check size

;disk parameter block for drive 3:
FE6D          DPB3:  DEFS 15

;translate table for 8inch single density drives (IBM-format):
FE7C 00 06 0C 12 XLT:  DEFB 0,6,12,18,24,4,10,16,22,2,8,14,20
FE80 18 04 0A 10
FE84 16 02 08 0E
FE88 14
FE89 01 07 0D 13      DEFB 1,7,13,19,25,5,11,17,23,3,9,15,21
FE8D 19 05 0B 11
FE91 17 03 09 0F
FE95 15

```

```

;table of ROM-addresses:
0000      $RESET EQU 0000H      ;jump to reset
00FE      $VDINIT EQU 00FEH     ;initialize the video controller chip M6845
010F      $RSINIT EQU 010FH     ;initialize the RS-232-C interface
0134      $KBCHAR EQU 0134H     ;get a keyboard character if available
0240      $KBWAIT EQU 0240H     ;wait for a keyboard character
02A5      $VDCHAR EQU 02A5H     ;display a character
0588      $PRSTAT EQU 0588H     ;test printer status
058D      $PRCHAR EQU 058DH     ;output a character to the printer
0597      $RSRCST EQU 0597H     ;get a character from the RS-232-C interface if
                                ;available
05BD      $RSRCV EQU 05BDH     ;receive a character from the RS-232-C
                                ;interface
05CA      $RSTXST EQU 05CAH     ;test the RS-232-C output status
05D9      $RSTX EQU 05D9H       ;transmit a character to the RS-232-C interface
05E3      $READ EQU 05E3H       ;read a disk sector
05EA      $WRITE EQU 05EAH      ;write a disk sector
07B7      $GETTIM EQU 07B7H     ;get time and date in binary format
07D4      $SETTIM EQU 07D4H     ;set time and date in binary format

;call ROM-I/O-driver:
00FA      $BSEL EQU 0FAH       ;bank select port

FE96      C5                  $ROMIO: PUSH BC      ;save vector number
FE97      D9                  EXX                ;save register set
FE98      C1                  POP BC             ;restore vector number
FE99      21 FEC6             LD HL,ROMIO     ;jump vector --> reg. HL
FE9C      06 00              LD B,0
FE9E      09                  ADD HL,BC   ;calculate jump address
FE9F      09                  ADD HL,BC
FEA0      4E                  LD C,(HL)   ;jump address --> reg. BC
FEA1      23                  INC HL
FEA2      46                  LD B,(HL)
FEA3      ED 43 FEB6          LD (CALLIO+1),BC ;store it
FEA7      ED 73 FEC4          LD (SAVE$P),SP ;save stack pointer
FEAB      31 F7E5             LD SP,HSTBUF-2 ;temporary stack --> stack pointer
FEAE      08                  EX AF,AF'   ;save accu
FEAF      3E F0              LD A,0F0H
FEB1      F3                  DI          ;disable interrupts
FEB2      D3 FA              OUT ($BSEL),A ;select ROM
FEB4      08                  EX AF,AF'   ;restore accu
FEB5      CD 0000             CALLIO: CALL 0000H ;call selected I/O-routine
FEB8      08                  EX AF,AF'   ;save accu
FEB9      3E FF              LD A,0FFH
FEBB      D3 FA              OUT ($BSEL),A ;deselect ROM
FEBD      08                  EX AF,AF'   ;restore accu
FEBE      ED 7B FEC4          LD SP,(SAVE$P) ;restore stack pointer
FEC2      D9                  EXX        ;restore register set
FEC3      C9                  RET         ;return to caller
FEC4      0000             SAVE$P: DEFW 0 ;temporary memory for stack pointer

```

```

;table of ROM-addresses:
ROMIO: DEFW $RESET,$VDINIT,$RSINIT,$KBCHAR,$KBWAIT,$VDCHAR,$PRSTAT,$PRCHAR

FEC6 0000 00FE
FECA 010F 0134
FECE 0240 02A5
FED2 0588 058D
FED6 0597 058D
FEDA 05CA 05D9
FEDE 05E3 05EA
FEE2 07B7 07D4
FEE6 074D

DEFW $RSRCST,$RSRCV,$RSTXST,$RSTX,$READ,$WRITE,$GETTIM,$SETTIM,074DH
```

```

;*****
;*                               S Y S T A B                               *
;*****

;RAM portion of I/O-drivers

;table of video parameters:
VIDPAR: DEFB 102      ;horizontal total
           DEFB 80      ;horizontal displayed
           DEFB 88      ;h sync position
           DEFB 52      ;h sync width
           DEFB 25      ;vertical total
           DEFB 0       ;v total adjust
           DEFB 24      ;vertical displayed
           DEFB 25      ;v sync position
           DEFB 0       ;interlace mode (non-interlace)
           DEFB 11      ;max scan line address
           DEFB 0       ;cursor start
           DEFB 11      ;cursor end
           DEFB 0       ;start address (H)
           DEFB 0       ;start address (L)
           DEFB 0       ;cursor (H)
           DEFB 0       ;cursor (L)

FEEB 66
FEE9 50
FEEA 58
FEEB 34
FEEC 19
FEED 00
FEEF 18
FEF0 19
FEF1 00
FEF2 08
FEF3 08
FEF4 00
FEF5 00
FEF6 00
FEF7 00

;RS-232-C interface initialization table:
FEF8 03      RS232C: DEFB 00000011B      ;bit 7, 6: don't care
                                           ;bit 5: This bit is the Stick Parity bit. When
                                           ; bit 3 is a logic 1 and bit 5 is a logic
                                           ; 1, the Parity bit is transmitted and
                                           ; checked by the receiver as a logic 0 if
                                           ; bit 4 is a logic 1 or as a logic 1 if
                                           ; bit 4 is a logic 0.
                                           ;bit 4: This bit is the Even Parity Select bit.
                                           ; When bit 3 is a logic 1 and bit 4 is a
                                           ; logic 0, an odd number of logic 1s is
                                           ; transmitted or checked in the data word
                                           ; bits and Parity bit. When bit 3 is a
                                           ; logic 1 and bit 4 is a logic 1, an even
                                           ; number of logic 1s is transmitted or
                                           ; checked.
                                           ;bit 3: This bit is the Parity Enable bit. When
                                           ; bit 3 is a logic 1, a Parity bit is
                                           ; generated (transmit data) or checked
                                           ; (receive data) between the last data
                                           ; word bit and Stop bit of the serial
                                           ; data (The parity bit is used to produce
                                           ; an even or odd number of 1s when the
                                           ; data word bits and the Parity bit are
                                           ; summed).

```

```

;bit 2: This bit specifies the number of Stop
; bits in each transmitted character. If
; bit 2 is a logic 0, one Stop bit is
; generated in the transmitted data. If
; bit 2 is a logic 1 when a 5-bit word
; length is selected via bits 0 and 1,
; one and a half Stop bits are generated.
; If bit 2 is a logic 1, when either 6-,
; 7-, or 8-bit word length is selected,
; two Stop bits are generated. The
; Receiver checks the first Stop-bit
; only, regardless of the number of Stop
; bits selected.
;bits 1, 0: These two bits specify the number
; of bits in each transmitted or
; received serial character. The
; encoding of bits 0 and 1 is as
; follows:
; 0 = 5 bits word length
; 1 = 6 " " "
; 2 = 7 " " "
; 3 = 8 " " "

```

FEF9 0014

DEFW 20

;divisor for a rate of 9600 Baud

;divisor table:

Desired Baud Rate	Divisor
50	3840
75	2560
110	1745
134.5	1428
150	1280
300	640
600	320
1200	160
1800	107
2000	96
2400	80
3600	53
4800	40
7200	27
9600	20
19200	10
38400	5

```

;keyboard driver:
FEFB 0898 DELAY1: DEFW 2200 ;determines wait time until key autorepeats
FEFD 0480 DELAY2: DEFW 1200 ;determines debounce time
; (keep it longer for bad keyboards)
FEFF 1388 DELAY3: DEFW 5000 ;determines frequency of key repeat

FF01 00 GERMAN: DEFB 0 ;switch bit to alter character set
; (0 = US-ASCII, FF = German)

FF02 0000 VECTOR: DEFW 0 ;function key vector

;table contains codes for all caps (may be altered by user):
FF04 5D 7D 61 41 KEYTAB: DEFM 'l}aAbBcCdDeEfgG' ;keybord address 3801H
FF08 62 42 63 43
FF0C 64 44 65 45
FF10 66 46 67 47
FF14 68 48 69 49 DEFM 'hHiIjJkKlLmMnNoO' ; 3802H
FF18 6A 4A 6B 4B
FF1C 6C 4C 6D 4D
FF20 6E 4E 6F 4F
FF24 70 5D 71 51 DEFM 'pPq@rRsStTuUvVwW' ; 3804H
FF28 72 52 73 53
FF2C 74 54 75 55
FF30 76 56 77 57
FF34 78 58 7A 5A DEFM 'xXzZyY;+[{:*@`' ; 3808H
FF38 79 59 3B 2B
FF3C 5B 7B 3A 2A
FF40 40 60
FF42 00 00 DEFB 0,0
FF44 30 5F 31 21 DEFM '0_1!2"3#4$5%6&7'''; 3810H
FF48 32 22 33 23
FF4C 34 24 35 25
FF50 36 26 37 27
FF54 38 28 39 29 DEFM '8(9)""\|, <=>./?' ; 3820H
FF58 5E 7E 5C 7C
FF5C 2C 3C 2D 3D
FF60 2E 3E 2F 3F
FF64 0D 0D 7F 1A DEFB CR, CR, DEL, SUB ; 3840H
FF68 1B FD 0B 12 DEFB ESC, OFDH, VT, DC2
FF6C 0A 03 0B 18 DEFB LF, ETX, BS, CAN, FF, HT
FF70 0C 09
FF72 20 20 DEFM ' '
FF74 80 80 81 81 DEFB 80H, 80H, 81H, 81H ; 38A0H
FF78 82 82 83 83 DEFB 82H, 82H, 83H, 83H
FF7C 84 84 85 85 DEFB 84H, 84H, 85H, 85H
FF80 86 86 87 87 DEFB 86H, 86H, 87H, 87H
FF84 3D 3D 31 31 DEFM '0011223344556677' ; 38C0H
FF88 32 32 33 33
FF8C 34 34 35 35
FF90 36 36 37 37
FF94 38 38 39 39 DEFM '8899' ; 38E0H
FF98 FE FE 00 00 DEFB OFEH, OFEH, 0, 0
FF9C 2C 2C 2D 2D DEFM ',,--..'
FFA0 2E 2E

```

```

;table of screen constants:
FFA2 18      VDTAB: DEFB 24      ;maximum line count
FFA3 50      DEFB 80      ;maximum column count
FFA4 00      DEFB 0       ;top line
FFA5 00      DEFB 0       ;left column
FFA6 17      DEFB 23      ;bottom line
FFA7 4F      DEFB 79      ;right column
FFA8 18      DEFB 24      ;line count
FFA9 50      DEFB 80      ;column count
FFAA 00      DEFB 0       ;current line
FFAB 00      DEFB 0       ;current column

;table of escape codes:
FFAC 0C      TAB1S: DEFB FF      ;cursor off
FFAD 0D      DEFB CR      ;cursor on
FFAE 3D      TAB2S: DEFB '='     ;absolute cursor addressing
FFAF 49      DEFB 'I'     ;set top line
FFB0 4A      DEFB 'J'     ;set bottom line
FFB1 4B      DEFB 'K'     ;set left column
FFB2 4C      TAB2E: DEFB 'L'   ;set right column
FFB3 50      DEFB 'P'     ;insert character
FFB4 51      DEFB 'Q'     ;delete character
FFB5 52      DEFB 'R'     ;inverse display mode
FFB6 53      DEFB 'S'     ;normal display mode
FFB7 56      DEFB 'V'     ;insert line
FFB8 57      TAB1E: DEFB 'W'   ;delete line

;table of control codes:
FFB9 07      TAB3S: DEFB BEL    ;beep
FFBA 08      DEFB BS      ;cursor left
FFBB 0A      DEFB LF      ;cursor down
FFBC 0B      DEFB VT      ;cursor up
FFBD 0C      DEFB FF      ;cursor right
FFBE 0D      DEFB CR      ;new line
FFBF 16      DEFB CAN     ;erase to end of line
FFC0 19      DEFB EM      ;erase to end of screen
FFC1 1A      DEFB SUB     ;clear screen
FFC2 1E      TAB3E: DEFB RS   ;home cursor

```

```

;drive control tables

;drive 0:
FFC3  70      DCT:  DEFB 01110000B      ;bit 7: size          0 = five inch floppy
;                                       ;                    1 = eight inch floppy
;bit 6: no. of surfaces 0 = single sided
;                                       ;                    1 = double sided
;bit 5: density        0 = single density
;                                       ;                    1 = double density
;bit 4: density of    0 = single density
;   first track      1 = double density
;bit 3: no. of first  0 = zero
;   sector on track  1 = one
;bit 2: steps per track 0 = one step
;   to track        1 = two steps
;bits 1,0: track stepping rate
;5 inch:  0 = 6 msec      1 = 12 msec
;          2 = 20 msec    3 = 30 msec
;8 inch:   0 = 3 msec     1 = 6 msec
;          2 = 10 msec    3 = 15 msec

FFC4  14      DEF8 20      ;sector count per track
FFC5  50      DEF8 80      ;number of usable tracks

FFC6  84      DEF8 10000100B ;bit 7,6: sector length
; 0 = 128 Bytes      1 = 256 Bytes
; 2 = 512 Bytes      3 = 1024 Bytes
;bits 5-1: interleaving factor (0 means 32)
;bit 0: drive status (used by disk driver --
;                do not alter)

FFC7  00      DEF8 0       ;current track

;drive 1:
FFC8  70      DEF8 01110000B
FFC9  14      DEF8 20
FFCA  50      DEF8 80
FFCB  84      DEF8 10000100B
FFCC  00      DEF8 0

;drive 2:
FFCD  8A      DEF8 10001010B
FFCE  1A      DEF8 26
FFCF  4D      DEF8 77
FFD0  02      DEF8 00000010B
FFD1  00      DEF8 0

;drive 3:
FFD2  60 14 50 42 DEF8 01100000B,20,80,01000010B,0
FFD6  00

FFD7  0A      TRYS: DEF8 10      ;# of times to try I/O until routine gives up
FFD8  F7E7    BUFFER: DEFW HSTBUF ;I/O buffer
FFDA  00      DRIVE: DEF8 0      ;drive
FFDB  0000    TRACK: DEFW 0      ;track
FFDD  00      SECTOR: DEF8 0     ;sector

FFDE  0000    TIMADR: DEFW 0     ;^(time & date)

```



;next variables should not be altered by user:

```

;keyboard:
FFED 00 00 00 00      KEYHLD: DEFB 0,0,0,0,0,0,0,0,0;keyboard work area
FFE4 00 00 00 00
FFE8 00 00
FFEA 0000             COUNT1: DEFW 0                ;delay counters for auto repeat function
FFEC 0000             COUNT2: DEFW 0
FFEE 0000             OLDROW: DEFW 0                ;temporary memory for keyboard driver
FFF0 00              OLCLMN: DEFB 0
FFF1 00              COUNT3: DEFB 0
FFF2 0000            FPTR: DEFW 0
FFF4 00              RDYKEY: DEFB 0                ;key code memory
FFF5                  DEFS 0                ;reserved for future expansion

;video:
FFF5 00              ESCAPE: DEFB 0                ;contains escape remarks
FFF6 00              ESCCHR: DEFB 0                ;memory for escape char
FFF7 00              LINE: DEFB 0                  ;memory for line number
FFF8 00              CURSOR: DEFB 0                ;0 = cursor on, OFFH = cursor off
FFF9 3C00            CURADR: DEFW 3C00H            ;memory for absolute cursor address
FFFB 00              INVERSE:DEFB 0                ;0 = normal mode, 80H = inverse mode

;disk:
FFFC 00              OLDDRV: DEFB 0                ;contains # of previous selected disk
FFFD 00              TRIES: DEFB 0                 ;counter for counting # of times to try I/O
                                           ;after error occurs
FFFE 00              RDFLAG: DEFB 0                ;flag (0 = WRITE, 1 = READ)

```

END

## Macros:

SMASK

## Symbols:

\$\$READ	F5A5	\$\$WRIT	F5BC	\$BOOT	F7E7	\$BSEL	00FA
\$CONIN	F4D8	\$CONOU	F4F6	\$CONST	F4B4	\$GETTI	07B7
\$HOME	F56D	\$KBCHA	0134	\$KBMAI	0240	\$LIST	F514
\$LISTS	F6EF	\$PRCHA	058D	\$PRSTA	0588	\$PUNCH	F530
\$READ	05E3	\$READE	F545	\$RESET	0000	\$ROMIO	FE96
\$RSINI	010F	\$RSRCS	0597	\$RSRCV	05BD	\$RSTX	05D9
\$RSTXS	05CA	\$SECTR	F720	\$SELDS	F575	\$SETDM	F5A0
\$SETSE	F59B	\$SETTI	07D4	\$SETTR	F596	\$START	F48A
\$VDCHA	02A5	\$VDINI	00FE	\$VMODE	00F5	\$WBOOT	F433
\$WRITE	05EA	ax	0000	ay	0001	ADD	F801
ALLOC	F61E	ALVO	FD27	ALV1	FD59	ALV2	FDBB
ALV3	FDBD	AORIGI	F958	ASERIA	F95D	BATST	F4D3
BDOS	E606	BEL	0007	BOOT	F400	BORIGI	F8B9
BS	0008	BUFFER	FFD8	CALLIO	FEB5	CAN	0018
CBOOTI	F46D	CCP	DE00	CGTLEN	003D	CHKROM	F887
CHKSER	F86D	CHKSUM	F0A1	CHKLUNA	F5E4	CLEAR	F83D
CLEAR1	F83E	CONIN	F409	CONOUT	F40C	CONST	F406
CONTB1	F4BD	CONTB2	F4E1	CONTB3	F4FF	CONVER	F8DC
COPYRG	427D	COUNT1	FFEA	COUNT2	FFEC	COUNT3	FFF1
CPMHST	FDEF	CR	000D	CRTIN	F4EE	CRTOUT	F50D
CRTSTI	F4C9	CRTSTO	F4D0	CSVD	FC67	CSV1	FC97
CSV2	FCC7	CSV3	FCF7	CURADR	FFF9	CURSOR	FFF8
DCT	FFC3	DECEND	F912	DECODE	F8AA	DEL	007F
DELAY1	FEFB	DELAY2	FEFD	DELAY3	FEFF	DI	00F3
DIRBUF	FBE7	DISPIN	F8CF	DISPVI	F82B	DMAADR	F6ED
DPBO	FE40	DPB1	FE4F	DPB2	FE5E	DPB3	FE6D
DPBASE	FE0D	DRIVE	FFDA	DSKVEC	F594	EM	0019
ENDMSG	FA70	EPROM	F8B8	ERFLAG	F6E9	ESC	001B
ESCAPE	FFF5	ESCCHR	FFF6	ETX	0003	FF	000C
FILHST	F665	FPTR	FFF2	GERMAN	FF01	HOME	F418
HSTACT	F6E2	HSTBUF	F7E7	HSTDSK	FFDA	HSTSEC	FFDD
HSTTRK	FFDB	HSTWRT	F6E3	HT	0009	ILLEN	000E
ILLMSG	F89C	INITMS	F912	INVERS	FFFB	IOBYTE	0003
JUMPIO	F714	KEYHLD	FFED	KEYTAB	FF04	LDERR	F492
LDSYS	F446	LF	000A	LINE	FFF7	LIST	F40F
LISTST	F42D	LOOP	F8EB	LOOP1	F8F7	LPTOUT	F52A
LPTST	F7D8	LPTST1	F712	LSTTB1	F522	LSTTB2	F6FC
MATCH	F682	MAXDRV	FDFD	MSG	F4AA	NOMATC	F65E
NOOVF	F618	NUL	0000	OLCLMN	FFF0	OLDDRV	FFFC
OLDROW	FFEE	ORIGIN	E128	OUTINI	F8BB	PTPOUT	F513
PTRIN	F4F3	PTRST	F4D0	PUNCH	F412	PUNTAB	F53D
RDFLAG	FFFE	RDRTB1	F550	RDRTB2	F558	RDYKEY	FFF4
READ	F427	READER	F415	READHS	F6D4	READOP	F6EB
RET	00C9	ROMIO	FEC6	RS	001E	RS232C	FEF8
RSFLAG	F6EA	RWMOVE	F6A7	RWOPER	F626	SAVESP	FEC4

SECTOR	FFDD	SECTRA	F430	SEKDSK	F6DD	SEKHST	F6E1
SEKSEC	F6ED	SEKTRK	F6DE	SELDISK	F41B	SERIAL	E12C
SETDMA	F424	SETSEC	F421	SETTRK	F41E	SKTRKC	F6BF
SUB	001A	TAB1E	FFB8	TAB1S	FFAC	TAB2E	FFB2
TAB2S	FFAE	TAB3E	FFC2	TAB3S	FFB9	TIMADR	FFDE
TRACK	FFDB	TRIES	FFFD	TRYS	FFD7	TTYIN	F4E9
TTYOUT	F507	TTYSTI	F4C5	TTYSTO	F7D4	UC1IN	F4F3
UC1OUT	F513	UC1ST	F4D0	UL1OUT	F513	UL1ST	F710
UNACNT	F6E4	UNADSK	F6E5	UNASEC	F6E8	UNATRK	F6E6
UR1IN	F4F3	UR1OUT	F513	UR1ST	F4D0	UR2IN	F4F3
UR2OUT	F513	UR2ST	F4D0	VDTAB	FFA2	VECTOR	FFD2
VIDPAR	FEE8	VIOLEN	0014	VIOMSG	F84C	VT	000B
WBOOT	F4D3	WRALL	0000	WRDIR	0001	WRITE	F42A
WRITEH	F6CB	WRTYPE	F6EC	WRUAL	0002	XLT	FE7C

Anlage B:

Window - Routine (in "C")

```

/*****
* WINDOW * ULIB002 * Thomas Holte * 831014 *
*****
*
*   ROUTINE FUER DIE EIN- BZW. AUSGABE   *
*   =====                           *
*
*   AUF EINEM BILDSCHIRMFENSTER       *
*   =====                           *
*
*
*   Version 2.1           (C) SoftCream GmbH           Thomas Holte *
*
*****/

```

```

char window (ctrl, anfang, ende, buffer)
  int ctrl, anfang, ende;
  char buffer[];

```

/\*

Diese Routine legt ueber einen Teil des Bildschirms ein Fenster beliebiger Groesse, von dem Daten eingelesen bzw. auf dem Daten ausgegeben werden koennen.

Bei der Dateneingabe koennen alle Editierfunktionstasten benutzt werden. Abgeschlossen wird die Routine durch den Druck einer der Funktionstasten F3-F6 bzw. der Taste BREAK. Die Daten werden dann in den buffer uebertragen, und es erfolgt die Rueckkehr ins aufrufende Programm.

Bedeutung der Uebergabeparameter:

ctrl = 16 Bit langer Kontrollcode mit nachstehender Bedeutung:

Bits 15-8: Hexadezimaler ASCII-Code eines evt. auszugebenden Promptzeichens. Sind diese Bits Null, wird das Blank als Prompt benutzt.

Bit 5: Piepston:

0 = aus

1 = an

Bit 4: Uebertragungsmodus:

0 = Abbruch durch Druck einer Funktionstaste (normaler Mode)

1 = Automatischer Abbruch bei Erreichen der letzten Kursorposition

Bits 3-0: Steuercode:

0 = Loeschen des Fensters (Clear window)

1 = Loeschen des Fensters und Eingabe (Clear, input)

3 = Ausgabe auf dem Fenster (Output )

4 = Editieren auf dem Fenster (Edit )

anfang = ZeilennummerSpaltennummer der ersten Zeichenposition des Fensters (siehe Bild 1).

Fuer diese beiden Angaben gilt: 0 <= Zeilennummer <= 23

0 <= Spaltennummer <= 79

Zeilennummer und Spaltennummer stehen hintereinander in einer int-Variablen.

ende = ZeilennummerSpaltennummer der letzten Spaltenposition des Fensters (siehe Bild 1).

buffer = Feld, in das bzw. aus dem die Daten gelesen werden.

Als Funktionswert wird der ASCII-Code der abschliessenden Taste zurueckgegeben.

```

|-----|
|Anfang |
|       |
|       |
|         W I N D O W
|         (Fenster)
|       |
|       |
|                         Ende
|-----|

```

Bild 1

\*/

```

#define TRUE  1
#define FALSE 0

```

```

#define CONIN 3

```

```

#define NUL 0x00
#define SOH 0x01
#define ACK 0x06
#define BEL 0x07
#define BS  0x08
#define HT  0x09
#define LF  0x0A
#define VT  0x0B
#define FF  0x0C
#define CR  0x0D
#define CAN 0x18
#define DEL 0x7F

```

{

```

char as, az, code, es, ez, flag, *init, ins, ls, lz, prompt, sanz, *setcur,
zanz;

```

```

ins = FALSE;           /* Vorbereiten des Insertflags */

```

```

code  = 0;
flag  = FALSE;
init  = "\33\14\33I \33J \33K \33L \36";
setcur = "\33= ";

```

```

/* Berechnen des Promptzeichens */

```

```

if ((prompt = ctrl >> 8) < ' ') prompt = ' ';

```

```

if (ctrl & 0x20) putchar (BEL);       /* Ausgabe eines Pieptons */

```

```

/* Pruefen, ob Flag fuer automatischen Abbruch gesetzt */

```

```

if (ctrl & 0x10) flag = TRUE;

```

```

/* Berechnen der ersten und letzten Zeile des Fensters */

```

```

init[4] = (az = anfang / 100) + ' ';
init[7] = (ez = ende   / 100) + ' ';

```

```

/* Berechnen der ersten und letzten Spalte des Fensters */

```

```

init[10] = (as = anfang % 100) + ' ';
init[13] = (es = ende   % 100) + ' ';

```

```

puts (init);

```

```

sanz = (es -= as) + 1;          /* Berechnen der Spaltenanzahl */
zanz = (ez -= az) + 1;          /* Berechnen der Zeilenanzahl */

switch (ctrl &= 7)
{
  case 1: setmem (buffer, zanz * sanz, ' ');

  case 0: for (lz = 0; lz <= ez; lz++)
    {
      for (ls = 0; ls <= es; ls++) putchar (prompt);
      if (lz < ez) putchar ('\n');
    }
    if (ctrl == 1) goto read;
    break;

  case 3:
  case 4: for (lz = 0; lz <= ez; lz++)
    {
      for (ls = 0; ls <= es; ls++) putchar (buffer[lz * sanz + ls]);
      if (lz < ez) putchar ('\n');
    }
    if (ctrl == 3) break;

  read:
  lz = ls = 0;
  for (;;)
  {
    setcur[2] = lz + ' ';
    setcur[3] = ls + ' ';
    puts (setcur); puts ("\33\i5");

    /* Holen eines Zeichens von der Tastatur */
    while (!(code = bdos(6, 0xFF)));
    puts ("\33\i4");

    if ((code &= 0x7F) < ' ')
    {
      ins = FALSE;
      switch (code)
      {
        case NUL: ins = TRUE;          /* ins mode */
                  continue;

        case SOH: if (lz < ez) movmem (&buffer[lz * sanz],
                                       /* ins line */ &buffer[(lz + 1) * sanz],
                                       (ez - lz) * sanz);
                  puts ("\33V");
                  setmem (&buffer[lz * sanz], sanz, ' ');
                  for (ls = 0; ls <= es; ls++) putchar (prompt);
                  ls = 0;
                  continue;

        case ACK: if (ls < es) movmem (&buffer[lz * sanz + ls + 1],
                                       /* del char */ &buffer[lz * sanz + ls ],
                                       es - ls);
                  puts ("\33Q");
                  setcur[3] = es + ' ';
                  puts (setcur);
                  buffer[lz * sanz + es] = ' ';
      }
    }
  }
}

```

```

        putchar (prompt);
        continue;

    case BEL: if (lz < ez) movmem (&buffer[(lz + 1) * sanz],
/* del line */      &buffer[ lz      * sanz],
                    (ez - lz) * sanz);

        puts ("\33W");
        setcur[2] = ez + ' ';
        setcur[3] = ' ';
        puts (setcur);
        setmem (&buffer[ez * sanz], sanz, ' ');
        for (ls = 0; ls <= es; ls++) putchar (prompt);
        ls = 0;
        continue;

    case BS : if (ls > 0) ls--; else if (!lz) break;
              continue;          /* cursor left */

    case HT : ls = es;            /* SHIFT + cursor right */
              continue;

    case LF : if (lz == ez) break; /* cursor down */
              lz++;
              continue;

    case VT : if (!lz) break;    /* cursor up */
              lz--;
              continue;

    case FF : if (ls < es) ls++; else if (lz == ez) break;
              continue;          /* cursor right */

    case CR : if (ez <= lz++) break; /* NEW LINE */
              ls = 0;
              continue;

    case CAN: ls = 0;            /* SHIFT + cursor left */
              continue;
    }
    break;
}

if (code == DEL)
{
    setcur[2] = setcur[3] = ' ';
    puts (setcur);
    setmem (buffer, sanz * sanz, ' ');
    for (lz = 0; lz <= ez; lz++)
    {
        for (ls = 0; ls <= es; ls++) putchar (prompt);
        if (lz < ez) putchar ('\n');
    }
    lz = ls = 0;
}
else
{
    if (ins)
    {
        if (ls < es) movmem (&buffer[lz * sanz + ls

```



```
                &buffer[lz * sanz + ls + 1], es - ls);
    puts ("\33P");
}
putchar (buffer[lz * sanz + ls] = code);
if (ls == es && lz == ez && flag) break;
if (++ls > es) ls = es;
}
}
}
puts ("\36\33I \33J\33K \33L\33\15");
return code;
}
```