```
                            ORG    :D23D
                    *
                    *
                    *
                    *  =============
                    *** I/O HANDLER ***
                    *  =============
                    *
                    *
                    *********************
                    * RUN basiccmd SAVE *
                    *********************
                    *
                    * Valid as direct command and in program.
                    * Clears the Heap, zeroes all variables, evaluate
                    * an evt. program name and writes a file of type
                    * 0 (BASIC) on tape.
                    *
                    * Exit: HL: Points to end symboltable.
                    *       DE: Length symboltable.
                    *       BC: Updated.
                    *
D23D CD23CB         RSAVE   CALL   :CB23     Empty HEAP + symtab
D240 2AA102                 LHLD   :02A1     Get start symtab
D243 EB                     XCHG             in DE
D244 2AA302                 LHLD   :02A3     End symtab in HL
D247 CD1ADE                 CALL   :DE1A     Calculate length symtab
D24A E5                     PUSH   H         Preserve length symtab
D24B 2A9F02                 LHLD   :029F     Get start textbuf
D24E EB                     XCHG             in DE
D24F CD1ADE                 CALL   :DE1A     Calculate length textbuf
D252 E5                     PUSH   H         Preserve length textbuf
D253 D5                     PUSH   D         Preserve start textbuf
D254 CD91E7                 CALL   :E791     (0) Evaluate program name
D257 00                     NOP
D258 00                     NOP
D259 00                     NOP
D25A 3E30                   MVI    A,:30     File type byte 0
D25C 00                     NOP
D25D 00                     NOP
D25E 00                     NOP
D25F 00                     NOP
D260 00                     NOP
D261 00                     NOP
D262 00                     NOP
D263 CDC502                 CALL   :02C5     Write fileleader, flagbyte,
                                             file type byte, name length
                                             and name.
D266 E1                     POP    H         Start textbuf in HL
D267 D1                     POP    D         Length textbuf in DE
D268 CDC802                 CALL   :02C8     Write length and contents
                                             textbuf
D26B D1                     POP    D         Length symtab in DE
D26C C3D8D7                 JMP    :D7D8     Write length and contents
                                             symtab + file trailer
                    *
D26F FF                     DATA   :FF
                    *
                    *********************
                    * RUN basiccmd LOAD *
                    *********************
                    *
```

```
                      * Valid as direct command and in program.
                      * Clears Heap and all variables. Evaluates name
                      * of program, updates BC. Required file type: 0.
                      * C is set to print type/name or not.
                      * A file (type 0: Basic) is read from tape. When
                      * a file has been found, the textbuffer and the
                      * symboltable are loaded and the pointers updated.
                      * When loading during program run: the program
                      * continues with the program just loaded.
                      *
                      * Exit: No error: BC: Updated.
                      *                 DE: Begin screen RAM.
                      *                 HL: End symbol table.
                      *
D270 CD23CB           RLOAD   CALL  :CB23     Empty HEAP + symtab
D273 CD91E7                   CALL  :E791     (0) Evaluate programname
D276 C5                       PUSH  B
D277 00                       NOP
D278 00                       NOP
D279 00                       NOP
D27A 00                       NOP
D27B 0630                     MVI   B,:30     File type byte 0
D27D E5                       PUSH  H         Preserve length name reqd
D27E 2A0001                   LHLD  :0100     Get CURRNT
D281 7C                       MOV   A,H
D282 B5                       ORA   L         Load during run program?
D283 0E00                     MVI   C,:00
D285 C289D2                   JNZ   :D289     If during run; C=00
D288 0D                       DCR   C         Else C=FF
D289 E1               RLD10   POP   H
D28A CDCE02                   CALL  :02CE     Switch on cassette motors;
                                              read header + name
D28D 2AA502                   LHLD  :02A5     Get end free RAM
D290 EB                       XCHG            Max. RAM in DE
D291 2A9F02                   LHLD  :029F     Start textbuf in HL
D294 CDD102                   CALL  :02D1     Load textbuffer
D297 22A102                   SHLD  :02A1     Store end textbuffer
D29A DCD102                   CC    :02D1     Load symboltable
D29D CD1AD7                   CALL  :D71A     Store end symtab; stop
                                              cassette motors.
D2A0 C1                       POP   B
D2A1 FB                       EI              Enable interrupts
D2A2 D2A8D2                   JNC   :D2A8     If loading error
D2A5 3E00                     MVI   A,:00     No loading error
D2A7 C9                       RET
                      *
                      *********************
                      * RUN LOADING ERROR *
                      *********************
                      *
                      * The programbuffers are restored. A error message
                      * is printed.
                      *
D2A8 F5               RLERR   PUSH  PSW
D2A9 CDB5DE                   CALL  :DEB5     Run 'NEW'
D2AC F1                       POP   PSW
D2AD 210000                   LXI   H,:0000
D2B0 220001                   SHLD  :0100     Set CURRNT=0
D2B3 C60B             RLEAR   ADI   :0B
D2B5 C3F5D9                   JMP   :D9F5     Run 'LOADING ERROR ..'
                      *
                      *
```

```
                   ******************
                   * OPEN TAPE FILE *
                   ******************
                   *
                   * Entry: A:  File type.
                   *        HL: Points to file name.
                   * Exit:  HL: Points beyond file name.
                   *        DE: Length of name.
                   *        BC: Preserved.
                   *        A:  Checksum on name.
                   *
D2B8 F5            CWOPEN  PUSH  PSW
D2B9 CD20D7                CALL  :D720     Init. write file leader
D2BC F1                    POP   PSW
D2BD CD09D5                CALL  :D509     Write file type byte
D2C0 C3F8D7                JMP   :D7F8     Get name length, write it
                                           on tape, incl. its c.s.
                   *
                   **********************
                   * RUN basiccmd CHECK *
                   **********************
                   *
                   * Valid as direct command only.
                   * Checks on file type and name. For all files
                   * with type <3, a checksum on all data is done.
                   * This routine remains in a endless loop and
                   * can be aborted with BREAK only.
                   *
                   RCHECK
D2C3 210000        CHK10   LXI   H,:0000   No program name given
D2C6 01FF00                LXI   B,:00FF   Any file type
D2C9 00                    NOP
D2CA 00                    NOP
D2CB CDCE02                CALL  :02CE     Read file header, file
                                           type and name; print
                                           type and name
D2CE 00                    NOP
D2CF FE33                  CPI   :33
D2D1 D2EBD2                JNC   :D2EB     If file type >=3: no check
                                           on checksum

                   * Test checksum::

D2D4 0C                    INR   C         BC=0
D2D5 CDE6D7                CALL  :D7E6     Set A=0, read + check
                                           a data block
D2D8 CCD702                CZ    :02D7     Read + check next block
D2DB C2E6D2                JNZ   :D2E6     If reading error
D2DE CDFFDA                CALL  :DAFF     Print 'OK', car.ret
D2E1 C0DB                  DBL   :DBC0
D2E3 C3C3D2                JMP   :D2C3     Wait for next file

                   * If checksum error:

D2E6 CDFFDA        CHK20   CALL  :DAFF     Print 'BAD'
D2E9 DBDB                  DBL   :DBDB
D2EB CD5EDD        CHK30   CALL  :DD5E     Print car.ret
D2EE C3C3D2                JMP   :D2C3     Wait for next file
                   *
                   ***********************
                   * WRITE BLOCK ON TAPE *
                   ***********************
```

```
                    *
                    * Entry: HL: Startaddres block.
                    *        DE: Length block.
                    * Exit:  HL: 1st byte after block.
                    *        A:  Checksum on block contents.
                    *        BCDE preserved.
                    *
D2F1 C5             CWBLK   PUSH  B
D2F2 D5                     PUSH  D
D2F3 00                     NOP
D2F4 CD16D3                 CALL  :D316      Write block length +
                                             c.s. on length
D2F7 0656                   MVI   B,:56      Initial checksum value
D2F9 7A             LD34    MOV   A,D
D2FA B3                     ORA   E
D2FB CA07D3                 JZ    :D307      If all bytes written
D2FE 1B                     DCX   D
D2FF 7E                     MOV   A,M        Get byte of block
D300 23                     INX   H          Point to next byte
D301 CD0FD3                 CALL  :D30F      Write byte, update checksum
D304 C3F9D2                 JMP   :D2F9      Next byte

                    * If all data written: write c.s. on block:

D307 78             LD35    MOV   A,B        Get calculated checksum
D308 CD09D5                 CALL  :D509      Write checksum
D30B 00                     NOP
D30C D1                     POP   D
D30D C1                     POP   B
D30E C9                     RET
                    *
                    ********************************
                    * WRITE BYTE, UPDATE CHECKSUM *
                    ********************************
                    *
                    * Entry: Byte to be written in A.
                    *        Checksum in B.
                    * Exit:  New checksum in B; A corrupted.
                    *        CDEHL preserved.
                    *
D30F CD09D5         LD36    CALL  :D509      Write byte
D312 A8                     XRA   B
D313 07                     RLC
D314 47                     MOV   B,A        Update checksum
D315 C9                     RET
                    *
                    *****************************************
                    * WRITE BLOCK LENGTH, UPDATE CHECKSUM *
                    *****************************************
                    *
                    * Entry: DE: length block.
                    * Exit:  DEHL preserved.
                    *
D316 0656           LD37    MVI   B,:56      Init checksum
D318 7A                     MOV   A,D        Get highest length byte,
D319 CD0FD3                 CALL  :D30F      write it, update c.s.
D31C 7B                     MOV   A,E        Get lowest length byte,
D31D CD0FD3                 CALL  :D30F      write it, update c.s
D320 78                     MOV   A,B        Get checksum
D321 CD09D5                 CALL  :D509      Write checksum on length
D324 C9                     RET
```

```
                ***********************
                * START FILE READING *
                ***********************
                *
                * Entry: HL: Address length byte of name requested.
                *        B:  File type byte requested.
                *        C:  00 when reading during run program,
                *            else FF.
                * Exit:  A:  File type byte.
                *        HL: Points to 1st byte of name requested.
                *        DE: Length name requested.
                *        BC: preserved.
                *
D325 F5         CROPEN  PUSH  PSW
D326 CDFFD7             CALL  :D7FF     Switch cassette motors on,
                                        init. registers
D329 00         RPN10   NOP
D32A 00                 NOP
D32B 00                 NOP
D32C 00                 NOP
D32D 00                 NOP
D32E F1                 POP   PSW
D32F CDF4D3             CALL  :D3F4     Read file header
D332 F5                 PUSH  PSW
D333 CD8AD7             CALL  :D78A     Display file type byte
D336 90                 SUB   B
D337 CDA2D3             CALL  :D3A2     Read and check header,
                                        program name, file type byte
D33A B7                 ORA   A         0 if everything OK.
D33B C283D7             JNZ   :D783     If failure
D33E F1                 POP   PSW
D33F C9                 RET
                *
                **************
                * READ BLOCK *
                **************
                *
                * Read length, contents and checksum of a block
                *
                * Entry: HL: Addr. where to dump data read.
                *        DE: End free space.
                * Exit:  CY=1: No error:
                *           HL: Next free address.
                *           BCDE preserved; AF corrupted.
                *        CY=0: Loading error:
                *           BCDEHL preserved.
                *           A: Type of loading error.
                *
D340 C5         CRBLK   PUSH  B
D341 D5                 PUSH  D
D342 E5                 PUSH  H
D343 CD90D7             CALL  :D790     Calculate free RAM space
D346 EB                 XCHG            Free RAM in DE
D347 CD8DD3             CALL  :D38D     Read block length + c.s.
                                        length in HL
D34A DA7ED3             JC    :D37E     If loading error 3
D34D B7                 ORA   A
D34E 3E00               MVI   A,:00     Loading error 0
D350 C280D3             JNZ   :D380     If checksum error 0
D353 E5                 PUSH  H         Save length block
D354 19                 DAD   D         Calculate free RAM
D355 D1                 POP   D         Get length block
```

```
D356 3C                 INR   A          Loading error 1
D357 E1                 POP   H
D358 E5                 PUSH  H          Restore begin addr.
D359 DA80D3             JC    :D380      If loading error 1
D35C 0656               MVI   B,:56      Init checksum
D35E 7A         LBK10   MOV   A,D
D35F B3                 ORA   E
D360 CA6FD3             JZ    :D36F      If whole block read
D363 1B                 DCX   D
D364 CD84D3             CALL  :D384      Read next byte, update c.s
D367 DA7ED3             JC    :D37E      If loading error 3
D36A 77                 MOV   M,A        Store byte in buffer
D36B 23                 INX   H
D36C C35ED3             JMP   :D35E      Next byte

                * If whole block read:

D36F CDD4D4     LBK20   CALL  :D4D4      Read checksum block contents
D372 DA7ED3             JC    :D37E      If loading error 3
D375 B8                 CMP   B          Check checksum
D376 3E02               MVI   A,:02      Loading error 2
D378 C280D3             JNZ   :D380      If loading error 2
D37B C3B4C6             JMP   :C6B4      CY=1, return: no error

                * If loading error:

D37E 3E03       LBK40   MVI   A.:03      Loading error 3
D380 B7         LOERR   ORA   A
D381 C3B6C6             JMP   :C6B6      Return with CY=0: error
                *
                **********************************
                * READ BYTE, CALCULATE CHECKSUM *
                **********************************
                *
                * Entry: B: Checksum.
                * Exit:  A: Byte read.
                *        B: Updated checksum.
                *        CDEHL preserved.
                *
D384 CDD4D4     INSC    CALL  :D4D4      Read byte
D387 F5         RBUEX   PUSH  PSW
D388 A8                 XRA   B          Calculate checksum
D389 07                 RLC
D38A 47                 MOV   B,A        Store new value
D38B F1                 POP   PSW
D38C C9                 RET
                *
                *******************
                * READ NAME LENGTH *
                *******************
                *
                * Entry: No conditions.
                * Exit:  HL: Length name read.
                *        A:  Result checksum check (0 if OK).
                *        BCDE: preserved.
                *        CY=0: O.K.; CY=1: Out of data.
                *
D38D C5         INLNG   PUSH  B
D38E 0656               MVI   B,:56      Init. checksum
D390 CD84D3             CALL  :D384      Read highest length byte
                                         and update checksum
D393 67                 MOV   H,A
```

```
D394 D484D3                CNC   :D384      Read lowest length byte
                                            and update checksum
D397 6F                    MOV   L,A
D398 D4D4D4                CNC   :D4D4      Read checksum on length
D39B F5            RHLEX   PUSH  PSW
D39C 90                    SUB   B          Check checksum
D39D 47                    MOV   B,A
D39E F1                    POP   PSW
D39F 78                    MOV   A,B
D3A0 C1                    POP   B
D3A1 C9                    RET
                   *
                   *******************************************
                   * READ + CHECK PROGRAM NAME AND FILE TYPE *
                   *******************************************
                   *
                   * Routine searches for proper file name by
                   * reading file name and compare it with name
                   * requested.
                   *
                   * Entry: A:  Evt. difference in file type byte
                   *            read and requested.
                   *        B:  Requested file type.
                   *        C:  00 during run program, else FF.
                   *        DE: Length requested.
                   *        HL: Address 1st byte name requested.
                   * Exit:  BCDEHL preserved.
                   *        A=0: All OK.
                   *        A=1: Loading error 1.
                   *
D3A2 C5            CMBLK   PUSH  B          Save file type + RUN flag
D3A3 E5                    PUSH  H          Save addr reqd name
D3A4 47                    MOV   B,A        Store deviation file type
D3A5 D5            MBK10   PUSH  D          Save req. name length
D3A6 E5                    PUSH  H
D3A7 CD8DD3                CALL  :D38D      Read + check program name
                                            evt. c.s.failure in A
D3AA DAEDD3                JC    :D3ED      If reading error
D3AD B7                    ORA   A
D3AE C2EDD3                JNZ   :D3ED      If checksum error
D3B1 E5                    PUSH  H          Save length name on tape
D3B2 CD1ADE                CALL  :DE1A      Calculate difference name
                                            lengths reqd and on tape
D3B5 7C                    MOV   A,H
D3B6 B5                    ORA   L
D3B7 67                    MOV   H,A        Difference in H
D3B8 68                    MOV   L,B        Difference file type in L
D3B9 D1                    POP   D          Get length name on tape
D3BA 0656                  MVI   B,:56      Initiate checksum
D3BC E3            MBK20   XTHL             Get byte reqd name
D3BD 7A                    MOV   A,D
D3BE B3                    ORA   E          Length name on tape = 0 ?
D3BF CAD8D3                JZ    :D3D8      If length = 0, or whole
                                            name read.
D3C2 1B                    DCX   D
D3C3 CD84D3                CALL  :D384      Read bytes of name, update
                                            checksum
D3C6 DAEDD3                JC    :D3ED      If reading error
D3C9 0D                    DCR   C
D3CA 0C                    INR   C          Load during run?
D3CB F5                    PUSH  PSW        Save length name on tape
D3CC C4EBD7                CNZ   :D7EB      Display program name
```

```
D3CF F1                  POP   PSW       Get byte of name on tape
D3D0 AE                  XRA   M         Compare with name reqd
D3D1 23                  INX   H
D3D2 E3                  XTHL            Get 'difference flag'
D3D3 B4                  ORA   H         Update it
D3D4 67                  MOV   H,A       and store it in H
D3D5 C3BCD3              JMP   :D3BC     Next byte

                * If whole name read:

D3D8 CDD4D4     MBK30    CALL  :D4D4     Read c.s on name contents
D3DB DAEDD3              JC    :D3ED     If reading error
D3DE A8         MBEX     XRA   B         Check checksum
D3DF E1                  POP   H
D3E0 B5                  ORA   L         Check file type
D3E1 6F                  MOV   L,A
D3E2 D1                  POP   D         Get length req. name
D3E3 7A                  MOV   A,D
D3E4 B3                  ORA   E         No name requested ?
D3E5 CAE9D3              JZ    :D3E9     If load without name
D3E8 7C                  MOV   A,H       Difference in names?
D3E9 B5         MBK40    ORA   L         Take also other checks in
                                         account
D3EA E1         MBK45    POP   H
D3EB C1                  POP   B
D3EC C9                  RET

                * If error:

D3ED E1         MBK50    POP   H
D3EE D1                  POP   D
D3EF 3E01                MVI   A,:01     Loading error 1
D3F1 C3E9D3              JMP   :D3E9
                *
                *
                *
D3F4                     END
```

```
***************************
* S Y M B O L   T A B L E *
***************************

CHK10   D2C3   CHK20   D2E6   CHK30   D2EB   CMBLK   D3A2
CRBLK   D340   CROPEN  D325   CWBLK   D2F1   CWOPEN  D2B8
INLNG   D38D   INSC    D384   LBK10   D35E   LBK20   D36F
LBK40   D37E   LD34    D2F9   LD35    D307   LD36    D30F
LD37    D316   LOERR   D380   MBEX    D3DE   MBK10   D3A5
MBK20   D3BC   MBK30   D3D8   MBK40   D3E9   MBK45   D3EA
MBK50   D3ED   RBUEX   D387   RCHECK  D2C3   RHLEX   D39B
RLD10   D2B9   RLEAR   D2B3   RLERR   D2AB   RLOAD   D270
RFN10   D329   RSAVE   D23D
```

```
                         ORG    :D3F4
                  *
                  *
                  *
                  *******************
                  * READ FILE HEADER *
                  *******************
                  *
                  * Locates a file on tape and reads leader.
                  * Exit: Interrupts are disabled. BCDEHL preserved.
                  *
D3F4 CD8FD9       RHDR   CALL   :D98F      Disable sound interrupt
D3F7 CD80D4              CALL   :D480      Find sync pattern
D3FA CDD4D4              CALL   :D4D4      Read flag type byte
D3FD DAF4D3              JC     :D3F4      Again if reading error
D400 FE55                CPI    :55
D402 C2F4D3              JNZ    :D3F4      Again if not flag byte
D405 CDD4D4              CALL   :D4D4      Read file type byte
D408 DAF4D3              JC     :D3F4      Again if reading error
D40B C9                  RET
                  *
                  ********************
                  * WRITE FILE LEADER *
                  ********************
                  *
                  * Writes a leader for program or data block on tape.
                  * Disables interrupts which could cause problems.
                  *
                  * Entry: at WHDR:  Entry if not during run program.
                  *        at WHD20: If during run of program.
                  * Exit:  BCDEHL preserved.
                  *
D40C CDFFDA       WHDR   CALL   :DAFF      Print 'SET RECORD, START
D40F 9CDB                DBL    :DB9C      TAPE, TYPE SPACE'
D411 CDCBD7              CALL   :D7CB      Wait for spacebar pressed
D414 CD2ED4       WHD20  CALL   :D42E      Switch on cassette motors
D417 F3                  DI                Disable interrupts
D418 00                  NOP
D419 00                  NOP
D41A CDEDD4              CALL   :D4ED      Write leader
D41D 3E55                MVI    A,:55      Get flag byte
D41F C309D5              JMP    :D509      Write flag byte
                  *
                  **************
                  * (Not used) *
                  **************
                  *
D422 CDF1D2       MPT27  CALL   :D2F1      Write block on tape
D425 00                  NOP
D426 00                  NOP
                  *
                  *********************
                  * WRITE FILE TRAILER *
                  *********************
                  *
                  * Write a trailer for program or datablock.
                  *
                  * Entry: Length of trailer in C.
                  * Exit:  A=0, BCDEHL preserved.
                  *
                  WTRL
D427 CD50D5       CWCLOS CALL   :D550      Write trailer bytes
```

```
D42A FB                   EI                  Enable interrupts
D42B C345D4               JMP    :D445        Stop cassette motors
                   *
                   ************************
                   * START CASSETTE MOTORS *
                   ************************
                   *
                   * Turns on motor of selected cassettedeck and
                   * waits 665 msec.
                   *
                   * Exit: All registers preserved.
                   *
D42E F5            CASST  PUSH   PSW
D42F 3A4000               LDA    :0040        Load POROM
D432 F630                 ORI    :30          Disable cassette motors
D434 E5                   PUSH   H
D435 213D01               LXI    H,:013D      Addr CASSL
D438 AE                   XRA    M            Get selected cassette
D439 E1                   POP    H
D43A 324000               STA    :0040        Remember POROM
D43D 3206FD               STA    :FD06        Switch cassette motor on
D440 CD41DE               CALL   :DE41        Delay
D443 F1                   POP    PSW
D444 C9                   RET
                   *
                   ***********************
                   * STOP CASSETTE MOTORS *
                   ***********************
                   *
                   * Switches off cassettemotors.
                   *
                   * Exit: All registers preserved.
                   *
                   CRCLOS
D445 F5            CASSP  PUSH   PSW
D446 3A4000               LDA    :0040        Load POROM
D449 F630                 ORI    :30          Disable cassette motors
D44B 324000               STA    :0040        Remember POROM
D44E 3206FD               STA    :FD06        Switch cassette motors off
D451 F1                   POP    PSW
D452 C9                   RET
                   *
                   ************
                   * READ BIT *
                   ************
                   *
                   * Reads one bit from tape.
                   *
                   * Entry: Address input port in HL. Input low state.
                   * Exit:  CY=0: sign bit of A is bit read.
                   *        CY=1: reading error.
                   *        EHL preserved.
                   *
D453 AF            RBIT   XRA    A
D454 57                   MOV    D,A
D455 47                   MOV    B,A
D456 4F                   MOV    C,A

                   * 1st impulse:

D457 05            RBT10  DCR    B
D458 CA7ED4               JZ     :D47E        Too long low
```

```
D45B B6                    ORA   M
D45C F257D4                JP    :D457      Wait for high
D45F 0D          RBT30     DCR   C
D460 CA7ED4                JZ    :D47E      Too long high
D463 15                    DCR   D
D464 A6                    ANA   M
D465 FA5FD4                JM    :D45F      Wait low
D468 010000                LXI   B,:0000

                 * 2nd impulse:

D46B 05          RBT40     DCR   B
D46C CA7ED4                JZ    :D47E      Too long low
D46F B6                    ORA   M
D470 F26BD4                JP    :D46B      Wait high again
D473 0D          RBT50     DCR   C
D474 CA7ED4                JZ    :D47E      Too long high
D477 14                    INR   D
D478 A6                    ANA   M
D479 FA73D4                JM    :D473      Wait low
D47C 7A                    MOV   A,D
D47D C9                    RET

                 * If error:

D47E 37          RBT90     STC              Set CY if error
D47F C9                    RET
                 *
                 ***************
                 * READ LEADER *
                 ***************
                 *
                 * Finds a section of leader on the tape.
                 *
                 * Entry: No conditions.
                 * Exit:  BCDEHL preserved, interrupts disabled.
                 *
D480 C5          RLEAD     PUSH  B
D481 D5                    PUSH  D
D482 E5                    PUSH  H
D483 0628                  MVI   B,:28      Estimate of impulse length
D485 2100FD                LXI   H,:FD00    address input port
D488 3EFF        RDL05     MVI   A,:FF
D48A FB          RDL10     EI               Enable interrupts
D48B 00                    NOP              (here cursor flashes)
D48C F3                    DI               Disable interrupts
D48D A6                    ANA   M
D48E FA8AD4                JM    :D48A      Wait low
D491 48                    MOV   C,B        Estimated length in C
D492 1614                  MVI   D,:14      Needs this many cycles for
                                            synchronisation. Must be
                                            more than trailer length.
D494 1E00        RDL30     MVI   E,:00
D496 AF                    XRA   A
D497 1D          RDL40     DCR   E
D498 CA88D4                JZ    :D488      Too long low; start again
D49B B6                    ORA   M
D49C F297D4                JP    :D497      Wait high
D49F 0600                  MVI   B,:00
D4A1 04          RDL50     INR   B
D4A2 CA88D4                JZ    :D488      Too long high; start again
D4A5 A6                    ANA   M
```

```
D4A6 FAA1D4                JM    :D4A1       Wait low
D4A9 78                    MOV   A,B
D4AA 91                    SUB   C           Compare impulse length
                                             with estimate
D4AB F2B0D4                JP    :D4B0
D4AE 2F                    CMA               )
D4AF 3C                    INR   A           ) 2-complement if <0
D4B0 5F           RDL60    MOV   E,A         Store difference in E
D4B1 79                    MOV   A,C         Calculate margin
D4B2 E6F0                  ANI   :F0
D4B4 1F                    RAR
D4B5 1F                    RAR
D4B6 1F                    RAR               Margin: 1/8th of estimate
D4B7 BB                    CMP   E           Compare with difference
D4B8 DAC3D4                JC    :D4C3       Not within margin

                  * If sync archieved:

D4BB 15                    DCR   D
D4BC C294D4                JNZ   :D494       Next impulse until D=0
D4BF 14                    INR   D
D4C0 C394D4                JMP   :D494       Next impulse until out
                                             of margin

                  * If out of margin:

D4C3 15           RDL70    DCR   D
D4C4 C288D4                JNZ   :D488       Not synchronised; again
D4C7 AF                    XRA   A
D4C8 B6           RDL80    ORA   M
D4C9 F2C8D4                JP    :D4C8       Wait high
D4CC A6           RDL90    ANA   M
D4CD FACCD4                JM    :D4CC       Wait low
D4D0 E1                    POP   H
D4D1 D1                    POP   D
D4D2 C1                    POP   B
D4D3 C9                    RET
                  *
                  *************
                  * READ BYTE *
                  *************
                  *
                  * Reads one byte from tape.
                  *
                  * Entry: No conditions.
                  * Exit:  CY=0: Byte read in A.
                  *        CY=1: Some error.
                  *        BCDEHL preserved.
                  *
D4D4 C5           RBYTE    PUSH  B
D4D5 D5                    PUSH  D
D4D6 E5                    PUSH  H
D4D7 2100FD                LXI   H,:FD00     Address input port
D4DA 1EFE                  MVI   E,:FE
D4DC CD53D4       RBY10    CALL  :D453       Read bit
D4DF DAE9D4                JC    :D4E9       If reading error; CY=1
D4E2 17                    RAL
D4E3 7B                    MOV   A,E
D4E4 17                    RAL
D4E5 5F                    MOV   E,A         Shift bit into E
D4E6 DADCD4                JC    :D4DC       Next bit
D4E9 E1           RBY20    POP   H           8 bits read, no error
```

```
D4EA D1                       POP    D
D4EB C1                       POP    B
D4EC C9                       RET
                    *
                    ****************
                    * WRITE LEADER *
                    ****************
                    *
                    * Writes a leader on the tape. From WLD10 also used
                    * to write a trailer.
                    *
                    * Entry: No conditions.
                    * Exit:  A=0, BCDEHL preserved.
                    *
D4ED 00             WLEAD     NOP
D4EE C5                       PUSH   B
D4EF E5                       PUSH   H
D4F0 2AE602                   LHLD   :02E6      Get leader impulse length
D4F3 01E807                   LXI    B,:07E8    Period for synchr.
D4F6 CD24D5         WLD10     CALL   :D524      Write bit
D4F9 0B                       DCX    B
D4FA 78                       MOV    A,B
D4FB B1                       ORA    C
D4FC C2F6D4                   JNZ    :D4F6      Write many bits
D4FF 2AE802                   LHLD   :02E8      Get impulse length data bit
D502 CD24D5                   CALL   :D524      Write a data '1' bit to end
D505 E1                       POP    H
D506 C1                       POP    B
D507 00                       NOP
D508 C9                       RET
                    *
                    **************
                    * WRITE BYTE *
                    **************
                    *
                    * Write a byte to tape.
                    *
                    * Entry: Byte to be written in A.
                    * Exit:  All registers preserved.
                    *
D509 F5             WBYTE     PUSH   PSW
D50A C5                       PUSH   B
D50B D5                       PUSH   D
D50C E5                       PUSH   H
D50D 2AE802                   LHLD   :02E8      Get impulse length bit '1'
D510 5C                       MOV    E,H
D511 55                       MOV    D,L        DE: inpulse length bit '0'
D512 0608                     MVI    B,:08      8 bits to write
D514 17             WBY10     RAL               Set/reset CY for kind of bit
D515 DC24D5                   CC     :D524      Write data '1' bit
D518 EB                       XCHG
D519 D424D5                   CNC    :D524      Write data '0' bit
D51C EB                       XCHG
D51D 05                       DCR    B
D51E C214D5                   JNZ    :D514      Next bit
D521 C356CB                   JMP    :CB56      Pop all, ret
                    *
                    *************
                    * WRITE BIT *
                    *************
                    *
                    * Write 2 impulses on tape, one long, one short.
```

```
                     *
                     * Entry: H: Half count first cycle.
                     *        L: Half count second cycle.
                     * Exit:  All registers preserved.
                     *
 D524 F5             WBIT     PUSH  PSW
 D525 D5                      PUSH  D
 D526 E5                      PUSH  H
 D527 6C                      MOV   L,H
 D528 1106FD                  LXI   D,:FD06   Address output port
 D52B CD3CD5                  CALL  :D53C     Write 1st impulse
 D52E E1                      POP   H
 D52F E5                      PUSH  H
 D530 65                      MOV   H,L
 D531 7D                      MOV   A,L
 D532 D608                    SUI   :08       Allow for return to WBYTE
 D534 6F                      MOV   L,A
 D535 CD3CD5                  CALL  :D53C     Write 2nd impulse
 D538 E1                      POP   H
 D539 D1                      POP   D
 D53A F1                      POP   PSW
 D53B C9                      RET
                     *
                     ***************
                     * WRITE CYCLE *
                     ***************
                     *
                     * Writes one impulse (hi/lo) on tape. Two cycles
                     * are required for one bit.
                     *
                     * Entry: DE: Address output port.
                     *        HL: Impulse length constants.
                     * Exit:  HL = 0. BCDE preserved.
                     *
 D53C 3A4000         WCYC     LDA   :0040     POROM in A
 D53F F601                    ORI   :01       lsb = 1
 D541 12                      STAX  D         Output port is made '1'
 D542 25             WCY10    DCR   H
 D543 C242D5                  JNZ   :D542     Write '1' until H=0
 D546 2D                      DCR   L
 D547 2D                      DCR   L
 D548 2D                      DCR   L         Allow for return to WBIT
 D549 3D                      DCR   A
 D54A 12                      STAX  D         Output port is made '0'
 D54B 2D             WCY20    DCR   L
 D54C C24BD5                  JNZ   :D54B     Write '0' until L=0
 D54F C9                      RET
                     *
                     ***********************
                     * WRITE TRAILER BITS *
                     ***********************
                     *
                     * Writes trailer bits after a block on tape.
                     *
                     * Entry: Number of trailer bits in C.
                     * Exit:  A=0, other registers preserved.
                     *        F corrupted.
                     *
 D550 C5             WTRLX    PUSH  B
 D551 E5                      PUSH  H
 D552 2AEA02                  LHLD  :02EA     Trailer impulse length
 D555 0600                    MVI   B,:00
```

```
D557 C3F6D4                  JMP    :D4F6      Write trailer bits
                    *
D55A FF                      DATA   :FF
D55B FF                      DATA   :FF
D55C FF                      DATA   :FF
D55D FF                      DATA   :FF
D55E FF                      DATA   :FF
D55F FF                      DATA   :FF
                    *
                    ********************************
                    * INITIALISE KEYBOARD POINTERS *
                    ********************************
                    *
                    * Set all keyboard pointers to default values.
                    *
                    * Entry: Address ASCII-table in HL (3E8C5).
                    * Exit:  BCDE preserved.
                    *
                    KLIRS
D560 22A702         KBINIT   SHLD   :02A7      Load pointer ASCII-table
D563 AF             KLIRP    XRA    A
D564 32B902                  STA    :02B9      Allow complete scan routine
D567 32C302                  STA    :02C3      CTRL not pressed
D56A 21BA02                  LXI    H,:02BA
D56D 22BE02                  SHLD   :02BE      Set KLIIN ) Ignore
D570 22C002                  SHLD   :02C0      Set KLIOU ) previous inputs
D573 2F                      CMA
D574 32C402                  STA    :02C4      BREAK pointer = FF
D577 C9                      RET
                    *
                    ***************************************
                    * KEYBOARD INTERRUPT SERVICE (RST 6) *
                    ***************************************
                    *
                    * Current interrupt mask is saved. Only stack and
                    * clock interrupts are allowed. Keyboard timer 4
                    * is re-loaded.
                    * KBXCT is counted down: abort if not 0.
                    * Else: scan keyboard and store result.
                    *
                    * Entry: None.
                    * Exit:  All registers + int. mask preserved.
                    *
D578 F5             KBINT    PUSH   PSW
D579 C5                      PUSH   B
D57A D5                      PUSH   D
D57B 3A5F00                  LDA    :005F
D57E F5                      PUSH   PSW        Preserve current int. mask
D57F 3E84                    MVI    A,:84      )
D581 32F8FF                  STA    :FFF8      ) Allow stack and clock
D584 325F00                  STA    :005F      ) interrupts only
D587 FB                      EI
D588 CD9DD9                  CALL   :D99D      Reload keyboard timer
D58B 21C101                  LXI    H,:01C1
D58E 35                      DCR    M          Decr. keyb.scan time count
D58F C2CDD9                  JNZ    :D9CD      No scanning if <>0

                    * if KBXCT = 0:

D592 3602                    MVI    M,:02      Set keyb. scan time counter
D594 CD9AD5                  CALL   :D59A      Scan keyboard, store result
D597 C3CDD9                  JMP    :D9CD      Restore int.mask; ret.
```

```
                        *
                        ******************************
                        * SCAN KEYBOARD, STORE RESULT *
                        ******************************
                        *
                        * Exit: All registers corrupted.
                        *
D59A 11F1FF             KBSCAN  LXI   D,:FFF1   Input port from keyboard
D59D 21F7FF                     LXI   H,:FFF7   Output port to keyboard
D5A0 01C402                     LXI   B,:02C4   BREAK pointer
D5A3 F3                         DI
D5A4 3640                       MVI   M,:40     Scan row 6
D5A6 1A                         LDAX  D         and get result
D5A7 FB                         EI
D5A8 87                         ADD   A         Check for BREAK pressed
D5A9 FA06D6                     JM    :D606     If BREAK pressed
D5AC CD50D7                     CALL  :D750     Update BREAK pointer
D5AF 3AB902                     LDA   :02B9     Get BREAK pointer
D5B2 B7                         ORA   A         Scan for BREAK only?
D5B3 C205D6                     JNZ   :D605     Then abort

                        * Scan all rows and store result in MAP1:

D5B6 01A902                     LXI   B,:02A9   MAP1 for currently
                                                pressed key
D5B9 C5                         PUSH  B         Preserve MAP1 addr
D5BA 3C                         INR   A         Determine row
D5BB F5                 KEB10   PUSH  PSW
D5BC F3                         DI
D5BD 77                         MOV   M,A       Scan row
D5BE 1A                         LDAX  D         Get result
D5BF FB                         EI
D5C0 02                         STAX  B         Store result in MAP1
D5C1 03                         INX   B
D5C2 F1                         POP   PSW
D5C3 87                         ADD   A         Determine next row
D5C4 D2BBD5                     JNC   :D5BB     Scan next row if not ready

                        * REPT handling:

D5C7 3AAF02                     LDA   :02AF
D5CA E620                       ANI   :20       Check if REPT pressed
D5CC 47                         MOV   B,A       Store result
D5CD 21C202                     LXI   H,:02C2   Addr. REPT counter
D5D0 7E                         MOV   A,M       Get contents
D5D1 3601                       MVI   M,:01     Update it for immediate scan
D5D3 CADFD5                     JZ    :D5DF     If REPT not pressed
D5D6 3D                         DCR   A         Else
D5D7 77                         MOV   M,A       Decr. REPT counter
D5D8 C204D6                     JNZ   :D604     If <>0, abort scan
D5DB 3602                       MVI   M,:02     Else RPCNT=2
D5DD 06FF                       MVI   B,:FF     Set B=FF for REPT pressed

                        * ASCII-value of key pressed into KLIND:

D5DF E1                 KEB40   POP   H         Get addr MAP1
D5E0 E5                         PUSH  H         Save addr. MAP1
D5E1 11B102                     LXI   D,:02B1   Get addr. MAP2
D5E4 0E00                       MVI   C,:00
D5E6 7E                 KEB50   MOV   A,M       Get result scan current
                                                row in A
D5E7 05                         DCR   B
```

```
D5E8 04                     INR   B        REPT pressed ?
D5E9 C2F0D5                 JNZ   :D5F0    If REPT pressed
D5EC EB                     XCHG
D5ED AE                     XRA   M        ) Check if new input
D5EE EB                     XCHG           )
D5EF A6                     ANA   M        )
D5F0 B7             KEB60   ORA   A        )
D5F1 C432D6                 CNZ   :D632    If new: Get ASCII-code
                                           and store it in KLIND
D5F4 13                     INX   D
D5F5 23                     INX   H        Next row
D5F6 0C                     INR   C
D5F7 79                     MOV   A,C
D5F8 FE08                   CPI   :08      All rows checked?
D5FA C2E6D5                 JNZ   :D5E6    Next row if not
D5FD D1             KEB70   POP   D        Get MAP1 addr in DE
D5FE D5                     PUSH  D
D5FF 44                     MOV   B,H      ) Get MAP2 addr in HL
D600 4D                     MOV   C,L      )
D601 CD4FDE                 CALL  :DE4F    Transfer (MAP1) into MAP2
D604 D1             KEB80   POP   D        Scrap
D605 C9             KEB81   RET

                    * if BREAK pressed:

D606 C5             KEB90   PUSH  B        Save Breakpntr
D607 CDA6D8                 CALL  :D8A6    All sound off
D60A C1                     POP   B
D60B 00                     NOP
D60C CD45D4                 CALL  :D445    Stop cassette motors
D60F 0A                     LDAX  B        Get KBRFL
D610 3C                     INR   A
D611 CA05D6                 JZ    :D605    If KBRFL=FF: break acknow-
                                           ledged already
D614 02                     STAX  B        Else: store new KBRFL
D615 FE20                   CPI   :20
D617 C205D6                 JNZ   :D605    If new KBRFL<>20: wait for
                                           soft-break to be accepted
D61A CD63D5                 CALL  :D563    Else: init keyb. pointers
D61D C30CC8                 JMP   :C80C    Print 'BREAK', return to
                                           monitor
                    *
                    *
D620                        END
```

```
***************************
* S Y M B O L   T A B L E *
***************************

CASSP   D445    CASST   D42E    CRCLOS D445    CWCLOS D427
KBINIT  D560    KBINT   D578    KBSCAN D59A    KEB10   D5BB
KEB40   D5DF    KEB50   D5E6    KEB60   D5F0   KEB70   D5FD
KEB80   D604    KEB81   D605    KEB90   D606   KLIRP   D563
KLIRS   D560    MPT27   D422    RBIT    D453   RBT10   D457
RBT30   D45F    RBT40   D46B    RBT50   D473   RBT90   D47E
RBY10   D4DC    RBY20   D4E9    RBYTE   D4D4   RDL05   D488
RDL10   D48A    RDL30   D494    RDL40   D497   RDL50   D4A1
RDL60   D4B0    RDL70   D4C3    RDL80   D4C8   RDL90   D4CC
RHDR    D3F4    RLEAD   D480    WBIT    D524   WBY10   D514
WBYTE   D509    WCY10   D542    WCY20   D54B   WCYC    D53C
WHD20   D414    WHDR    D40C    WLD10   D4F6   WLEAD   D4ED
WTRL    D427    WTRLX   D550
```

```
                         ORG    :D620
                  *
                  *
                  *
                  **************************
                  * COMPLETE KEYBOARD SCAN *
                  **************************
                  *
                  * Initialises a complete keyboard scan,
                  * independent of the KNSCAN flag, and performs it.
                  *
                  * Exit: All registers preserved.
                  *
D620 F5           KFSCAN PUSH   PSW
D621 C5                  PUSH   B
D622 D5                  PUSH   D
D623 E5                  PUSH   H
D624 21B902              LXI    H,:02B9   Addr KNSCAN pointer
D627 3600                MVI    M,:00     Enable complete scan
D629 E5                  PUSH   H
D62A CD9AD5              CALL   :D59A     Scan keyboard and store
                                          result in circ.buffer
D62D E1                  POP    H
D62E 35                  DCR    M         Scan for BREAK only
D62F C356CB              JMP    :CB56     Popall; return
                  *
                  ********************************
                  * GET ASCII VALUE OF KEY PRESSED *
                  ********************************
                  *
                  * Calculates address in ASCII table in ROM and gets
                  * ASCII value of the key pressed. The result is
                  * stored in the 4-bye circular buffer KLIND.
                  *
                  * Entry: A:  Keycode of scanned row (7 bits only).
                  *        B:  FF when REPT pressed; else 00.
                  *        C:  Number of row.
                  *        DE: Address in MAP2.
                  *        HL: Address in MAP1.
                  * Exit:  BCDEHL preserved; AF corrupted.
                  *
D632 C5           TKEY   PUSH   B
D633 0607                MVI    B,:07     Check which key in row is
D635 1F           TKY10  RAR              pressed; calculate offset
D636 DC3FD6              CC     :D63F     Get ASCII value; store it
                                          in KLIND
D639 05                  DCR    B
D63A C235D6              JNZ    :D635     Next column
D63D C1                  POP    B
D63E C9                  RET
                  *
                  * GET KEY-ASCII VALUE AND STORE IT:
                  *
D63F CF           SINKEY RST    1         Get ASCII-value from ROM-
D640 12                  DATA   :12       table and store it in KLIND
D641 C9                  RET
                  *
                  ******************************
                  * OUTPUT TO RS232 IF REQUIRED *
                  ******************************
                  *
                  * Checks if output is to RS232. If positive,
                   * output is performed.
                   *
                   * Entry: Byte to be transmitted in A.
                   *
D642 F5            TOUTSE  PUSH  PSW
D643 3A3101                LDA   :0131      Get output direction
D646 B7                    ORA   A          Check if RS232 output
D647 C28CDD                JNZ   :DD8C      Abort if not
D64A F1                    POP   PSW
D64B C394DD                JMP   :DD94      Output to RS232
                   *
                   ******************************************
                   * GET ASCII-VALUE OF CHARACTER IN BUFFER *
                   ******************************************
                   *
                   * Routine is not used.
                   *
                   * The Ascii-value of a character is stored in
                   * KLIND. Afterwards, Bank select is restored.
                   *
D64E CD3FE9        LD67    CALL  :E93F      (3) Ascii-value in KLIND
D651 F1                    POP   PSW        A contains POROM
D652 CD08D8                CALL  :D808      Update PORO and POROM
D655 F1                    POP   PSW
D656 E1                    POP   H
D657 C9                    RET
                   *
                   ************************************
                   * parts of RUN 'RANDOMISE' (0E40C) *
                   ************************************
                   *
                   RMI15
D658 3D            MPT39   DCR   A
D659 C258D6                JNZ   :D658      Again till A=0
D65C 7E                    MOV   A,M        Get contents FD00
D65D AB                    XRA   E
D65E C9                    RET
                   *
                   * Entry: L = 0.
                   *
D65F 3AC101        MPT38   LDA   :01C1      Get keyb.scan time count
                                            (0, 1 or 2)
D662 0F                    RRC
D663 0F                    RRC              A= 0, #40 or #80
D664 5F                    MOV   E,A        in E
D665 45                    MOV   B,L        )
D666 4D                    MOV   C,L        ) BC=0
D667 C9                    RET
                   *
                   ************************************
                   * WRITE 2 BLOCKS + TRAILER ON TAPE *
                   ************************************
                   *
                   * Entry: HL:    Startaddress 1st block.
                   *          Stack: Length 2nd block.
                   *                 Startaddress 2nd block.
                   *
D668 110100        MPT13   LXI   D,:0001    Length 1st block = 1
D66B CDCB02                CALL  :02CB      Write 1st block
D66E D1                    POP   D          Get length 2nd block
D66F E1                    POP   H          Get startaddr. 2nd block
D670 CDC802                CALL  :02C8      Write 2nd block
```

```
D673 CDCB02              CALL   :02CB      Write trailer
D676 B7                  ORA    A
D677 C9                  RET
                 *
                 ********************************
                 * LOADA: EVALUATE PROGRAM NAME *
                 ********************************
                 *
                 * The program name is evaluated. Selection of
                 * ROM bank 1 is prepared.
                 *
D678 CD87D6      MPT14   CALL   :D687      Evaluate program name
D67B 3A4000              LDA    :0040      Get POROM
D67E F640                ORI    :40        Prepare selection ROMbank 1
D680 C9                  RET
                 *
                 ******************
                 * OPEN READ FILE *
                 ******************
                 *
D681 D5          MPT18   PUSH   D
D682 CDCE02              CALL   :02CE      Open READ file
D685 D1                  POP    D
D686 C9                  RET
                 *
                 *********************************
                 * EVALUATE A STRING EXPRESSSION *
                 *********************************
                 *
                 * A string expression is evaluated. Eventually,
                 * the Heap entry is cleared if the string was
                 * temporarily on Heap.
                 *
                 * Exit: DE preserved, BC updated.
                 *       HL points after string
                 *
D687 D5          MPT15   PUSH   D
D688 CD91E7              CALL   :E791      (0) Eval. string expr.
                                           evt. clear Heap entry
D68B D1                  POP    D
D68C C9                  RET
                 *
                 *******************
                 * CURSOR HANDLING *
                 *******************
                 *
                 * Load the cursor pointers with the address, the
                 * colour and the contents of a new cursor address.
                 *
                 * Entry: HL: New cursor address.
                 *         D:  The colour byte of this location.
                 *         E:  The contents of this location.
                 * Exit:  HL and DE exchanged; ABCF preserved.
                 *
D68D 227200      SPT00   SHLD   :0072      Store cursor address
D690 EB                  XCHG
D691 227600              SHLD   :0076      Store cursor addr.contents
D694 C9                  RET
                 *
                 ************************
                 * OUTPUT ONE CHARACTER *
                 ************************
```

```
                        *
                        * Output direction depending on OTSW.
                        * This routine is useable for all data output
                        * functions in machine language programs.
                        *
                        * Entry: Character for output in A.
                        * Exit:  All registers preserved.
                        *
    D695 F5             MPT31   PUSH  PSW
    D696 CD60DD                 CALL  :DD60     Output character in A.
    D699 F1                     POP   PSW
    D69A C9                     RET
                        *
    D69B FF                     DATA  :FF
                        *
                        ************************************************
                        * KEYB. SCANNING: UPDATE POINTER OUTPUT BUFFER *
                        ************************************************
                        *
                        * Updates the pointer to the circular output
                        * buffer #02BA-#02BD.
                        *
                        * Entry: HL: KLIOU.
                        * Exit:  HL: Updated KLIOU.
                        *        AF corrupted. BCDE preserved.
                        *
    D69C 23             KPTRU   INX   H         Incr. KLIOU
    D69D 7D                     MOV   A,L       Lobyte into A
    D69E FEBE                   CPI   :BE       Buffer full?
    D6A0 C0                     RNZ             Quit if not
    D6A1 21BA02                 LXI   H,:02BA   Else: wrap around
    D6A4 C9                     RET
                        *
                        ******************************************
                        * KEYBOARD SCANNING: CHECK IF NEW INPUTS *
                        ******************************************
                        *
                        * Returns a flag if BREAK has been pressed or if
                        * there is a character available.
                        *
                        * Entry: No conditions.
                        * Exit:  BCDEHL preserved.
                        *        A: Difference KLIIN and KLIOU.
                        *        CY=1: Break pressed.
                        *        CY=0, Z=1: No inputs.
                        *        CY=0, Z=0: New input available.
                        *
                        ASKKEY
    D6A5 E5             BREAK   PUSH  H

                        * If suspended:

    D6A6 21C402                 LXI   H,:02C4   Addr break pntr
    D6A9 7E                     MOV   A,M
    D6AA 3D                     DCR   A
    D6AB FEFE                   CPI   :FE       Test if not 0 or FF
    D6AD DAB9D6                 JC    :D6B9     Abort if break: CY=1
                        *
    D6B0 2AC002                 LHLD  :02C0     Get KLIOU
    D6B3 3ABE02                 LDA   :02BE     Get KLIIN
    D6B6 95                     SUB   L         New keys pressed?
    D6B7 37                     STC
```

```
D6B8 3F                     CMC                CY=0
D6B9 E1            OTK10    POP    H
D6BA C9                     RET
                   *
                   ******************
                   * INPUT SCANNING *
                   ******************
                   *
                   * Gets input from keyboard or DINC, depending on
                   * INSW (0296).
                   *
                   * FGETC: Gets a character, even if keyboard
                   *        scanning is turned off.
                   *  GETC: Returns a flag if break, and sets break
                   *        accepted. Returns also a flag if a
                   *        character is available.
                   *
                   * Exit: CY=1: Break pressed.
                   *       Z=1:  No inputs. Then A=0.
                   *       Else: Character in A.
                   *
D6BB CD20D6        FGETC    CALL   :D620       Complete keyboard scan
D6BE C3DBD1        GETC     JMP    :D1DB       Check input keyb/DINC;
                                               scan for new keys pressed
D6C1 DAD4D6        MPR29    JC     :D6D4       Jump if Break pressed.
D6C4 C8                     RZ                 No new input or buffer full

                   * If inputs:

D6C5 E5                     PUSH   H
D6C6 2AC002                 LHLD   :02C0       Get addr pntr output buffer
D6C9 7E                     MOV    A,M         Get ASCII char in A
D6CA F5                     PUSH   PSW
D6CB CD9CD6                 CALL   :D69C       Update pntr
D6CE F1                     POP    PSW
D6CF 22C002                 SHLD   :02C0       Re-instate KLIOU
D6D2 E1                     POP    H
D6D3 C9                     RET                CY=0

                   * If Break pressed:

D6D4 3EFF          GTC10    MVI    A,:FF       Flag 'break accepted'
D6D6 32C402                 STA    :02C4       Scan for break only
D6D9 C9                     RET                CY=1
                   *
                   *********************
                   * WAIT FOR SPACEBAR *
                   *********************
                   *
                   * Wait until spacebar (or break) is pressed.
                   *
                   * Entry: None.
                   * Exit:   CY=1: Break pressed.
                   *         CY=0: Space in A.
                   *         BCDEHL preserved.
                   *
D6DA CDBBD6        WSPACE   CALL   :D6BB       Input scanning
D6DD D8                     RC                 Abort if BREAK pressed
D6DE FE20                   CPI    :20         Check if spacebar
D6E0 C2DAD6                 JNZ    :D6DA       Wait for space bar
D6E3 B7                     ORA    A
D6E4 C9                     RET
```

```
                *
                *************************
                * part of LOADA (1EE0F) *
                *************************
                *
D6E5 E3         MPT20   XTHL            Orig. DE in HL, free RAM
                                        pntr on stack
D6E6 CD14DE             CALL  :DE14     Compare DE-HL
D6E9 D2EDD6             JNC   :D6ED     If DE<=HL
D6EC EB                 XCHG
D6ED 42         RLA15   MOV   B,D       ) Lowest value in BC
D6EE 4B                 MOV   C,E       )
D6EF D1                 POP   D
D6F0 E1                 POP   H
D6F1 E3                 XTHL
D6F2 C34CEE             JMP   :EE4C     (1) Continu
                *
                *****************************************
                * CHECK SUFFICIENT SCREEN RAM AVAILABLE *
                * PREPARE SELECTION SPLIT MODE          *
                *****************************************
                *
D6F5 37         SPT01   STC             CY=1
D6F6 2A9000             LHLD  :0090     Get end area splitting mode
D6F9 CDA6E5             CALL  :E5A6     (2) Ask for temporary area
D6FC 21A0E5             LXI   H,:E5A0   (2) Startaddr table screen
                                        parameters split modes
D6FF C9                 RET
                *
                ******************************************
                * CHANGE FROM SPLIT TO FULL GRAPHIC MODE *
                ******************************************
                *
D700 CDF5D6     SSM20   CALL  :D6F5     Check suff. screen RAM
D703 C385E4             JMP   :E485     (2) Change split to full
                *
                *****************************************
                * CHECK SUFFICIENT SCREEN RAM AVAILABLE *
                * PREPARE FULL GRAPHICS MODE            *
                *****************************************
                *
D706 CDF5D6     SPTA2   CALL  :D6F5     Check suff. screen RAM
D709 219AE5             LXI   H,:E59A   (2) Startaddr table screen
                                        parameters full graph.modes
D70C C9                 RET
                *
                ******************************
                * SET UP CURRENT SCREEN MODE *
                ******************************
                *
D70D D1         SMA20   POP   D
D70E F1                 POP   PSW
D70F 3A9D00     SMA30   LDA   :009D     Get current screen mode
D712 B7                 ORA   A
D713 1F                 RAR             Split or all-graph mode ?
D714 CD39E5             CALL  :E539     (2) Set up screen mode
D717 C33CE4             JMP   :E43C     (2) Pop PSW, ret
                *
                *************************
                * STOP LOADING PROGRAMS *
                *************************
                *
```

```
                    * Entry: HL: New end symbol table.
                    * Exit:  All registers preserved.
                    *
D71A 22A302         MPT11   SHLD  :02A3      Store end symtab
D71D C3D402                 JMP   :02D4      Stop loading
                    *
                    ****************************
                    * INIT. WRITING FILE LEADER *
                    ****************************
                    *
                    * Procedure depends on saving in program or not.
                    *
D720 E5             MPT21   PUSH  H
D721 2A0001                 LHLD  :0100      Get start current line
D724 7C                     MOV   A,H
D725 B5                     ORA   L          0 if not during run
D726 E1                     POP   H
D727 C214D4                 JNZ   :D414      If SAVE during run
D72A C30CD4                 JMP   :D40C      Write file leader
                    *
                    ******************************************
                    * INIT. SOUNDGENERATOR, GIC, START HEAP, *
                    * MOVE CASSETTE VECTORS, GET DCE INPUTS  *
                    ******************************************
                    *
D72D CDA6D8         MPT00   CALL  :D8A6      Init. sound generator
D730 CD95D7                 CALL  :D795      Transfer cassette vectors
D733 21EC02                 LXI   H,:02EC
D736 CF                     RST   1          Init. GIC; get evt. inputs
D737 0C                     DATA  :0C        from DCE-bus (bootstrap)
D738 229B02                 SHLD  :029B      Set HEAP start
D73B C9                     RET
                    *
D73C 02                     DATA  :02        (not used)
                    *
                    *******************************************
                    * part of RUN A VARIABLE REFERENCE (0E95A) *
                    *******************************************
                    *
D73D CD6DE9         MPT49   CALL  :E96D      (0) Run VARPTR
D740 D1                     POP   D
D741 C9                     RET
                    *
D742 DD                     DATA  :DD        (not used)
                    *
                    ***************************************************
                    * SET INPUT DIRECTION, LOAD SOUND + KEYB TIMERS *
                    ***************************************************
                    *
                    * Part of 'stack interrupt' (D9E2).
                    *
D743 323501         MPT30   STA   :0135      Set input direction
D746 CD9DD9                 CALL  :D99D      Reload keyboard timer
D749 C3A3D9                 JMP   :D9A3      Reload sound timer, ret
                    *
                    ******************************
                    * DATA OUTPUT ROUTINE 'DOUTC' *
                    ******************************
                    *
                    * Part of DD70.
                    * On #02DD, an jump to an user determined output
                    * routine can be written. As default, a RET is
```

```
                      * on this address.
                      *
D74C F1               OTC30   POP   PSW        Output char in A
D74D C3DD02                   JMP   :02DD      Goto user DOUTC
                      *
                      ********************
                      * CHECK BREAK FLAG *
                      ********************
                      *
                      * Part of 'scan keyboard' (D59A).
                      *
                      * Entry: Address 'break' flag in BC.
                      * Exit:  BCDEHL preserved, AF corrupted.
                      *
D750 0A               MPT28   LDAX  B          Get KBRFL
D751 3C                       INR   A
D752 C0                       RNZ              Quit if it was <> FF
D753 02                       STAX  B          KBRFL=0: No recent break
D754 C9                       RET
                      *
                      ***************************
                      * SOUND INTERRUPT (RST 3) *
                      ***************************
                      *
                      * Called periodically every few milliseconds.
                      * Adjust the volume for the sound channels and
                      * approaches the correct frequency if necessary.
                      *
                      * Saves all registers + interrupt mask.
                      * Enables only clock and sound interrupts.
                      * Sound interrupt timer is re-loaded and sound
                      * control blocks are executed.
                      *
                      * Entry: HL must already be saved on stack.
                      * Exit:  All registers preserved.
                      *        Bank select is restored.
                      *
D755 F5               SNTMP   PUSH  PSW
D756 C5                       PUSH  B
D757 D5                       PUSH  D
D758 3A5F00                   LDA   :005F      Get current int. mask
D75B F5                       PUSH  PSW        and save it
D75C 3E84                     MVI   A,:84      )
D75E 325F00                   STA   :005F      ) Enable clock and sound
D761 32F8FF                   STA   :FFF8      ) interrupts only
D764 FB                       EI
D765 CDA3D9                   CALL  :D9A3      Reload sound timer
D768 3A4000                   LDA   :0040      Get POROM
D76B F5                       PUSH  PSW        and save it
D76C E63F                     ANI   :3F
D76E F640                     ORI   :40        Select ROM bank 1
D770 324000                   STA   :0040      Set POROM
D773 3206FD                   STA   :FD06      and PORO
D776 CD6EEE                   CALL  :EE6E      (1) Execute SCB('s)
D779 F1                       POP   PSW
D77A 324000                   STA   :0040      Re-instate old ROM bank
D77D 3206FD                   STA   :FD06      and save it
D780 C3CDD9                   JMP   :D9CD      Restore int. mask; ret
                      *
                      *************************
                      * FAILURE DURING ROPEN *
                      *************************
```

```
                   *
D783 05            MPT54   DCR   B
D784 04                    INR   B
D785 C2DED7                JNZ   :D7DE     If file type byte <>0:
                                           Run error
D788 F1                    POP   PSW
D789 C9                    RET
                   *
                   **********************************
                   * CHECK IF LOAD DURING RUN PROGRAM *
                   **********************************
                   *
                   * Entry: C: 00 if load during run, else it is FF.
                   *        A: File type byte.
                   * Exit:  ABCDEHL preserved.
                   *
D78A 0D            MPT24   DCR   C
D78B 0C                    INR   C         Check C
D78C C8                    RZ              Abort if during run
D78D C3EBD7                JMP   :D7EB     Display file type byte
                   *
                   *
                   *
D790                       END
```

```
****************************
* S Y M B O L   T A B L E *
****************************
```

ASKKEY	D6A5	BREAK	D6A5	FGETC	D6BB	GETC	D6BE
GTC10	D6D4	KFSCAN	D620	KPTRU	D69C	LD67	D64E
MPR29	D6C1	MPT00	D72D	MPT11	D71A	MPT13	D668
MPT14	D678	MPT15	D687	MPT18	D681	MPT20	D6E5
MPT21	D720	MPT24	D78A	MPT28	D750	MPT30	D743
MPT31	D695	MPT38	D65F	MPT39	D658	MPT49	D73D
MPT54	D783	OTC30	D74C	OTK10	D6B9	RLA15	D6ED
RMI15	D658	SINKEY	D63F	SMA20	D70D	SMA30	D70F
SNTMP	D755	SPT00	D68D	SPT01	D6F5	SPTA2	D706
SSM20	D700	TKEY	D632	TKY10	D635	TOUTSE	D642
WSPACE	D6DA						

```
                                ORG    :D790
                        *
                        *
                        *
                        ************************
                        * CHECK FREE RAM SPACE *
                        ************************
                        *
                        * Entry: DE: Startaddress.
                        *        HL: 1st not useable address.
                        * Exit:  HL: Useable RAM space.
                        *        ABCDE preserved.
                        *
    D790 CD1ADE         MPT25   CALL   :DE1A     Calculate free space
    D793 2B                     DCX    H
    D794 C9                     RET
                        *
                        **********************************
                        * TRANSFER DATA/CASSETTE VECTORS *
                        **********************************
                        *
                        * Transfer data/cassette switching vectors from
                        * ROM to RAM vector area.
                        *
                        * Exit: AFBC preserved. DEHL corrupted.
                        *
                        MPT01
    D795 C5             CASIN   PUSH   B
    D796 21CBD7                 LXI    H,:D7CB   Highest source address
    D799 11A4D7                 LXI    D,:D7A4   Lowest source address
    D79C 01C502                 LXI    B,:02C5   Lowest destination address
    D79F CD4FDE                 CALL   :DE4F     Transfer cassette vectors
    D7A2 C1                     POP    B
    D7A3 C9                     RET
                        *
                        ***********************************
                        * DATA/CASSETTE SWITCHING VECTORS *
                        ***********************************
                        *
                        * This data block is moved into the RAM area
                        * #02C5-#02EB during system initialisation.
                        *
    D7A4 C3B8D2         CINTB   JMP    :D2B8     WOPEN
    D7A7 C3F1D2                 JMP    :D2F1     WBLK
    D7AA C327D4                 JMP    :D427     WCLOSE
    D7AD C325D3                 JMP    :D325     ROPEN
    D7B0 C340D3                 JMP    :D340     RBLK
    D7B3 C345D4                 JMP    :D445     RCLOSE
    D7B6 C3A2D3                 JMP    :D3A2     MBLK
    D7B9 C9                     RET              RESET
    D7BA 00                     NOP
    D7BB 00                     NOP
    D7BC C9                     RET              DOUTC
    D7BD 00                     NOP
    D7BE 00                     NOP
    D7BF C3B4DD                 JMP    :DDB4     DINC
    D7C2 C9                     RET
    D7C3 00                     NOP
    D7C4 00                     NOP
    D7C5 2424                   DATA   :24,:24   TAPSL
    D7C7 243C                   DATA   :24,:3C   TAPSD
    D7C9 2418                   DATA   :24,:18   TAPST
```

```
                    *
                    **********************************
                    * WAIT FOR SPACEBAR, PRINT CAR.RET *
                    **********************************
                    *
                    * Exit: BCDEHL preserved.
                    *       CY=1: Break pressed.
                    *
                    CINTE
D7CB CDDAD6         WPT     CALL  :D6DA     Wait for spacebar
D7CE DA0CC8                 JC    :C80C     If BREAK pressed: into
                                            Basic monitor
D7D1 C35EDD                 JMP   :DD5E     Print car.ret; ret
                    *
D7D4 FF                     DATA  :FF
D7D5 FF                     DATA  :FF
D7D6 FF                     DATA  :FF
D7D7 FF                     DATA  :FF
                    *
                    ******************************
                    * WRITE BLOCK + TRAILER ON TAPE *
                    ******************************
                    *
                    * Entry: DE: Length block.
                    *        HL: Startaddress block.
                    * Exit:  A=0, BCDE preserved.
                    *        HL points past block written.
                    *
D7D8 CDC802         MPT10   CALL  :02C8     Write block
D7DB C3CB02                 JMP   :02CB     Write trailer
                    *
                    **********************
                    * FAILURE DURING ROPEN *
                    **********************
                    *
D7DE 0D             RPN20   DCR   C
D7DF 0C                     INR   C         Load during program ?
D7E0 C45EDD                 CNZ   :DD5E     Print car.ret if not
D7E3 C329D3                 JMP   :D329     Back to read file leader
                    *
                    ************************
                    * CHECK FILE OF ANY TYPE *
                    ************************
                    *
                    MPT12
D7E6 3E00           AMBLK   MVI   A,:00     File type correct
D7E8 C3D702                 JMP   :02D7     Read and check program name
                    *
                    **************************************
                    * WRITE BYTE ON CURSOR POSITION ADDRESS *
                    * AND UPDATE CURSOR POSITION.           *
                    **************************************
                    *
                    * This routine is a fast printing routine:
                    * the data byte is poked directly into the
                    * screen RAM.
                    *
                    * Entry: Byte to be written on screen in A.
                    * Exit:  All registers preserved.
                    *
D7EB E5             LD95    PUSH  H
D7EC 00                     NOP
```

```
D7ED 2A7200             LHLD  :0072     Get cursor position address
D7F0 77                 MOV   M,A       Write byte on screen
D7F1 2B                 DCX   H         ) Update cursor addr
D7F2 2B                 DCX   H         )
D7F3 227200             SHLD  :0072     Save new cursor address
D7F6 E1                 POP   H
D7F7 C9                 RET
                *
                **************************
                * SAVE: WRITE NAME LENGTH *
                **************************
                *
                * Entry: HL: Addr. length byte of name.
                * Exit:  DE: Length of name.
                *        HL: Points past string.
                *        BC: Preserved.
                *        A:  Checksum on string.
                *
D7F8 5E         MPT22   MOV   E,M       Get name length
D7F9 1600               MVI   D,:00
D7FB 23                 INX   H         HL to 1st byte of name
D7FC C3F1D2             JMP   :D2F1     Write name length
                *
                **********************************
                * INITIALISE LOADING FROM TAPE *
                **********************************
                *
                * Entry: HL: Points to length byte of name requested
                * Exit:  DE: Length requested name.
                *        HL: Points to first byte of name.
                *        AFBC preserved.
                *
D7FF 5E         MPT23   MOV   E,M       Get length requested name
D800 1600               MVI   D,:00     in DE
D802 23                 INX   H         HL points to 1st byte name
D803 C32ED4             JMP   :D42E     Cassette motors on; ret
                *
                *********************
                * UPDATE POROM/PORO *
                *********************
                *
                * Entry: MPT52: Byte for ROM/cassette select in A.
                *        MPT53: New POROM byte.
                *
D806 E6F0       MPT52   ANI   :F0       Enable ROM/cassette select
D808 324000     MPT53   STA   :0040     Load POROM
D80B 3206FD             STA   :FD06     and PORO
D80E C9                 RET
                *
                ****************
                * part of 2EAC1 *
                ****************
                *
                * If pointer is off top visible screen:
                *
D80F 7A         PCK40   MOV   A,D
D810 FED0               CPI   :D0
D812 DAEFEA             JC    :EAEF     (2) if <= CF (no overflow
                                        below 0)
D815 E5                 PUSH  H         ) if > CF:
D816 C5                 PUSH  B         ) Exchange BC and HL
D817 E1                 POP   H         )
```

```
D818 C1                  POP    B          )
D819 C3EFEA              JMP    :EAEF      (2)
                 *
D81C FF                  DATA   :FF
                 *
                 **********************
                 * RUN basiccmd SAVEA *
                 **********************
                 *
D81D CD3BD8      RSAVA   CALL   :D83B      Evaluate array type to be
                                           saved
D820 F5                  PUSH   PSW        Save type array
D821 C226D8              JNZ    :D826      Jump if INT/FPT array
D824 E7                  RST    4          Move stringarray into one
D825 75                  DATA   :75        string in free RAM
D826 F1          LD99    POP    PSW        Get type array
D827 E5                  PUSH   H
D828 D5                  PUSH   D
D829 F5                  PUSH   PSW
D82A CD91E7              CALL   :E791      (0) Evaluate program name
D82D 3E32                MVI    A,:32      File type byte '2'
D82F CDC502              CALL   :02C5      WOPEN
D832 F1                  POP    PSW        Get type array
D833 213E01              LXI    H,:013E    Startaddr EBUF
D836 77                  MOV    M,A        Type into EBUF
D837 C368D6              JMP    :D668      Write 2 blocks + trailer
                 *
D83A 00                  NOP
                 *
                 ****************************************
                 * EVALUATE ARRAY TYPE TO BE SAVED/LOADED *
                 ****************************************
                 *
                 * Exit: A:    Array type.
                 *       DE:   Length all array elements.
                 *       HL:   Beginaddr. 1st array element.
                 *       Z=1:  String array.
                 *       Z=0:  INT/FPT array.
                 *
D83B CD5AE9      RLSAS   CALL   :E95A      (0) Get array addr in HL
D83E E630                ANI    :30        Allow any type array
D840 F5                  PUSH   PSW        Save type
D841 5E                  MOV    E,M        )
D842 23                  INX    H          ) Get array pntr in DE
D843 56                  MOV    D,M        )
D844 7B                  MOV    A,E
D845 B2                  ORA    D
D846 CA90E9              JZ     :E990      (0) Undef. array if no addr
D849 EB                  XCHG              Pointer to array in HL
D84A 2B                  DCX    H
D84B 2B                  DCX    H
D84C 56                  MOV    D,M        Get 1st length byte
D84D 23                  INX    H
D84E 7E                  MOV    A,M        Get 2nd length byte
D84F 23                  INX    H
D850 96                  SUB    M          Minus nr of dim. bytes
D851 5F                  MOV    E,A
D852 7A                  MOV    A,D
D853 DE00                SBI    :00        Update hibyte if borrow
D855 57                  MOV    D,A
D856 1B                  DCX    D          DE is length all array elem.
D857 CD39DE              CALL   :DE39      HL is beginaddr 1st element
```

```
D85A F1                  POP    PSW       Get type in A
D85B FE20                CPI    :20       Set Z-flag on type
D85D C9                  RET
                *
                *********************
                * RUN basiccmd LOADA *
                *********************
                *
D85E CD3BD8     RLODA    CALL   :D83B     Evaluate array type to be
                                          loaded
D861 E5                  PUSH   H         Save startaddr array elem.
D862 F5                  PUSH   PSW       Save type
D863 CD78D6              CALL   :D678     Evaluate requested program
                                          name; prep. select ROM bank
D866 CD08D8              CALL   :D808     Select ROM bank 1
D869 F1                  POP    PSW       Get type
D86A C30FEE              JMP    :EE0F     (1) Read block from tape anc
                                          store it in array
                *
                **********************************
                * INITIALISE 'EDIT': EMPTY HEAP,    *
                * CLEAR SYMBOL TABLE, MOVE PROGRAM *
                **********************************
                *
                * Part of 'Init. EDIT' (0E265).
                *
                * Entry: CY=1: Not sufficient memory available.
                *
D86D DA10DA     MPT33    JC     :DA10     Evt. run error 'OUT OF
                                          MEMORY'
D870 C323CB              JMP    :CB23     Empty heap, move
                                          program, clear symtab
                *
                ******************************************
                * LIST CURRENT LINE IF LINENR IS CORRECT *
                ******************************************
                *
                * Part of 'run LIST <range>' (0E1B6).
                *
                * Entry: CY=0 if linenr. <= 0 or > FFFF.
                *
D873 D215DA     MPT3A    JNC    :DA15     Evt. run error 'NUMBER
                                          OUT OF RANGE'
D876 C3ABEC              JMP    :ECAB     (0) List current line
                *
                **************************
                * INPUT FROM EDIT BUFFER *
                **************************
                *
                * Part of 'restart interpreter' (C823).
                *
D879 E5         MPT05    PUSH   H
D87A CD91E2              CALL   :E291     (0) Input from editbuffer
D87D E1                  POP    H
D87E C9                  RET
                *
                ******************************
                * part of RUN 'CLEAR' (0E6B5) *
                ******************************
                *
                * Checks for heap too big.
                *
```

```
D87F CDF8E6      MPT42   CALL   :E6F8       (0) Get space req. in HL
D882 7C                  MOV    A,H
D883 B7                  ORA    A           Set flags on hibyte
D884 37                  STC                CY=1 if > 32K
D885 C9                  RET
                 *
                 ********************************
                 * EVT. INITIALISE 4 COLOUR ANIMATE *
                 ********************************
                 *
                 * Part of 2E9C3.
                 *
D886 FE14        SPT04   CPI    :14
D888 D0                  RNC                Abort if A >= 14
D889 32C100              STA    :00C1       Set for 4-colour animate
D88C C9                  RET
                 *
                 ********
                 * DATA *
                 ********
                 *
D88D 20          LD221   DATA   :20         Space
D88E 8C72                DATA   :8C,:72     Pntr. to 'MODE'
D890 00                  DATA   :00
                 *
                 ***************************
                 * part of 1EE0B - (not used) *
                 ***************************
                 *
D891 F5          LD105   PUSH   PSW
D892 CD91E7              CALL   :E791       (1)
D895 F1                  POP    PSW
D896 C9                  RET
                 *
                 *****************************************
                 * READ BLOCK FROM TAPE, EVT. ERROR REPORT *
                 *****************************************
                 *
                 * Part of LOADA (1EE0F).
                 *
D897 CDD102      MPT19   CALL   :02D1       Read block from tape
D89A D2B3D2              JNC    :D2B3       Evt. run 'LOADING ERROR'
D89D C9                  RET
                 *
                 ************************************
                 * LIST ARRAY NAME, SPACE, EXPRESSION *
                 ************************************
                 *
                 * Used in listing 'Savea/Loada' textlines.
                 *
D89E CDF7EE      SCN30   CALL   :EEF7       (0) List array name
D8A1 C365ED              JMP    :ED65       (0) List space, expression
                 *
D8A4 FF                  DATA   :FF
D8A5 FF                  DATA   :FF
                 *
                 ***************************
                 * INITIALISE SOUND GENERATOR *
                 ***************************
                 *
                 * All sound channels are switched off.
                 *
```

```
                  * Exit: AB preserved, CDEHLF corrupted.
                  *
D8A6 2106FC       SNDINI  LXI    H,:FC06   )
D8A9 3636                 MVI    M,:36     ) Load 3 timers
D8AB 3676                 MVI    M,:76     )
D8AD 36B6                 MVI    M,:B6     )
D8AF 210000               LXI    H,:0000
D8B2 2204FD               SHLD   :FD04     Volume 4 channels off
D8B5 229402               SHLD   :0294     and remember it
D8B8 21C201               LXI    H,:01C2   Start 1st SCB
D8BB 110E00               LXI    D,:000E   Length SCB
D8BE 0E04                 MVI    C,:04     4 blocks (3 SCB, 1 NCB)
D8C0 36FF         SNI10   MVI    M,:FF     FF in 1st byte SCB (= off)
D8C2 19                   DAD    D         Calc start next block
D8C3 0D                   DCR    C
D8C4 C2C0D8               JNZ    :D8C0     Next block if not ready
D8C7 C9                   RET
                  *
                  ********************
                  * OUTPUT TO DCE-BUS *
                  ********************
                  *
                  * 'Real World' output. Writes a byte to a given
                  * Real World address.
                  *
                  * Entry: D: Busaddress.
                  *        E: Data for output.
                  *
D8C8 F5           RWOP    PUSH   PSW
D8C9 E5                   PUSH   H
D8CA 2103FE               LXI    H,:FE03   GIC control address
D8CD 3680                 MVI    M,:80     All ports output
D8CF 2B                   DCX    H         Port C addr. in HL
D8D0 36FE                 MVI    M,:FE     Clear bus expand signal
D8D2 EB                   XCHG             Data in L, busaddr. in H
D8D3 2200FE               SHLD   :FE00     Data in PA, busaddr. in PB
D8D6 EB                   XCHG             Address PC in HL
D8D7 34                   INR    M         Set bus expand signal
D8D8 36FD                 MVI    M,:FD     Set write strobe true
                                           (Now data exchange done)
D8DA 36FF                 MVI    M,:FF     Reset strobe
D8DC 35                   DCR    M         Clear bus expand signal
D8DD E1                   POP    H
D8DE F1                   POP    PSW
D8DF C9                   RET
                  *
                  *********************
                  * INPUT FROM DCE-BUS *
                  *********************
                  *
                  * 'Real World' input. Reads a byte from a given
                  * Real World address.
                  *
                  * Entry: D: Busaddress.
                  * Exit:  E: Data received.
                  *
D8E0 F5           RWIP    PUSH   PSW
D8E1 E5                   PUSH   H
D8E2 2103FE               LXI    H,:FE03   GIC control addr. in HL
D8E5 3690                 MVI    M,:90     PA input, rest output
D8E7 2B                   DCX    H         Address PC in HL
D8E8 36FE                 MVI    M,:FE     Clear bus expand signal
```

```
D8EA 7A                          MOV   A,D      Busaddress in A
D8EB 3201FE                      STA   :FE01    Store busaddress in PB
D8EE 34                          INR   M        Set bus expand signal
D8EF 36FB                        MVI   M,:FB    Set read strobe true
                                                (Now data exchange)
D8F1 3A00FE                      LDA   :FE00    Data to A
D8F4 5F                          MOV   E,A      Data in E
D8F5 36FF                        MVI   M,:FF    Reset strobe
D8F7 35                          DCR   M
D8F8 E1                          POP   H
D8F9 F1                          POP   PSW
D8FA C9                          RET
                     *
                     *
                     *
D8FB                             END

***************************
* S Y M B O L   T A B L E *
***************************

AMBLK  D7E6   CASIN  D795   CINTB  D7A4   CINTE  D7CB
LD105  D891   LD221  D88D   LD95   D7EB   LD99   D826
MPT01  D795   MPT05  D879   MPT10  D7D8   MPT12  D7E6
MPT19  D897   MPT22  D7F8   MPT23  D7FF   MPT25  D790
MPT33  D86D   MPT3A  D873   MPT42  D87F   MPT52  D806
MPT53  D808   PCK40  D80F   RLODA  D85E   RLSAS  D83B
RPN20  D7DE   RSAVA  D81D   RWIP   D8E0   RWOP   D8CB
SCN30  D89E   SNDINI D8A6   SNI10  D8C0   SPT04  D886
WPT    D7CB
```