```
                        **********************
                        * COMPARE TWO STRINGS *
                        **********************
                        *
                        * Compares two string character by character.
                        * No characters >=#80 are allowed.
                        *
                        * Entry: DE: beginaddr. 1st string.
                        *        HL: beginaddr. 2nd string.
                        * Exit:  ABCDEHL preserved.
                        *        Flags:     Z=1: Both strings ident.
                        *                 S=1,Z=0: 2nd string longer.
                        *                 S=0,Z=0: 1st string longer.
                        *
D121 C5         SHCOMP  PUSH  B
D122 F5                 PUSH  PSW
D123 D5                 PUSH  D
D124 E5                 PUSH  H
D125 1A                 LDAX  D         Get length 1st string
D126 47                 MOV   B,A       Store it in B
D127 4E                 MOV   C,M       Length 2nd string in C
D128 13         LD3     INX   D
D129 23                 INX   H
D12A 78                 MOV   A,B
D12B D601               SUI   :01       Check 1st string empty
D12D 47                 MOV   B,A       Save rest of bytes
D12E DA45D1             JC    :D145     If 1st string empty
D131 79                 MOV   A,C
D132 D601               SUI   :01       Check 2nd string empty
D134 4F                 MOV   C,A       Save rest of bytes
D135 3C                 INR   A
D136 3C                 INR   A
D137 DA3FD1             JC    :D13F     If 2nd string empty
D13A 1A                 LDAX  D         Get byte 1st string
D13B BE                 CMP   M         and compare it with 2nd
D13C CA28D1             JZ    :D128     If identical: cont. test
D13F E1         LD4     POP   H
D140 D1                 POP   D
D141 C1                 POP   B
D142 78                 MOV   A,B       Restore A
D143 C1                 POP   B
D144 C9                 RET

                        * If 1st string empty:

D145 79         LD5     MOV   A,C       Get length 2nd string
D146 B7                 ORA   A
D147 CA3FD1             JZ    :D13F     If also empty: abort, Z=1
D14A AF                 XRA   A
D14B 3D                 DCR   A         Z=0
D14C C33FD1             JMP   :D13F     Quit
                        *
                        ****************************************
                        * EXTRACT A SUBSTRING FROM THE MIDDLE OF *
                        * ANOTHER STRING                         *
                        ****************************************
                        *
                        * Entry: HL points to string.
                        *        D  offset substring.
                        *        E  length substring.
                        * Exit:  If O.K.: HL points to substring.
                        *        Else: Jump to error.
```



```
                        *        AF corrupted. BCDE preserved.
                        *
D14F C5         SHMID   PUSH  B
D150 D5                 PUSH  D
D151 7E         SHM05   MOV   A,M       Get length string
D152 92                 SUB   D         Minus offset substring
D153 DA15DA             JC    :DA15     Run error 'NUMBER OUT OF
                                        RANGE' if offset too big
D156 93                 SUB   E         Minus length substring
D157 DA15DA             JC    :DA15     Run error 'NUMBER OUT OF
                                        RANGE' if length too big
D15A 7A         SHM08   MOV   A,D       Get offset
D15B CD30DE             CALL  :DE30     Calc beginaddr substring
D15E 23                 INX   H
D15F E5                 PUSH  H         Save begin substring
D160 7B                 MOV   A,E       Get length substring
D161 CD8BD1             CALL  :D18B     Find place in Heap for
                                        substring
D164 D1                 POP   D         Get begin substring
D165 E5                 PUSH  H
D166 CD73D1             CALL  :D173     Move substring into Heap
D169 E1                 POP   H
D16A D1                 POP   D
D16B C1                 POP   B
D16C C9                 RET
                        *
                        ****************************
                        * TRANSFER OF STRING DATA *
                        ****************************
                        *
                        * Copies strings from one place to another.
                        *
                        * Start at SCOPF: Transfer known string from DE
                        *                 to HL.
                        * Start at SCOPT: Transfer string into limited
                        *                 space of HL.
                        *
                        * Entry: DE: Startaddr. string to be transferred.
                        *        HL: Destination address.
                        *            1st bytes are length bytes.
                        * Exit:  Both DE + HL point to byte after string.
                        *        BC preserved; A=FF, CY=1.
                        *
D16D 1A         SCOPF   LDAX  D         Get length
D16E 13                 INX   D         Pnt to 1st byte
D16F C375D1             JMP   :D175
                        *
D172 13         SHCOPY  INX   D         Pnt to 1st byte of string
D173 7E         SCOPT   MOV   A,M       Get available space
D174 23                 INX   H         Pnt to 1st place to store
D175 C5         LD9     PUSH  B
D176 47                 MOV   B,A       Available space/space reqd
                                        in B
D177 78         LD10    MOV   A,B       Get space still available
D178 D601               SUI   :01       Decr. space available
D17A 47                 MOV   B,A       and save it
D17B DA85D1             JC    :D185     Jump if ready
D17E 1A                 LDAX  D         Get byte to be transferred
D17F 77                 MOV   M,A       and transfer it
D180 13                 INX   D         Pnt to next byte
D181 23                 INX   H         Pnt to next place
D182 C377D1             JMP   :D177     Transfer next byte
```

```
                        *
D185 C1          LD11   POP   B
D186 C9                 RET
                        *
                        ************************
                        * RELINGUISH HEAP SPACE *
                        ************************
                        *
                        * Clears a string in the heap.
                        *
                        * Entry: HL points to 2nd length byte of string.
                        * Exit:  HL points to 2nd length byte of cleared
                        *        string. AFBCDE preserved.
                        *
D187 2B          SHREL  DCX   H
D188 C336D2             JMP   :D236     Clear Heap entry.
                        *
                        ************************************
                        * REQUEST SPACE IN HEAP FOR A STRING *
                        ************************************
                        *
                        * Finds a place in the heap for a new string.
                        *
                        * Entry: A: Required space.
                        * Exit:  AFBCDE preserved.
                        *        HL points to length of reserved area.
                        *        Error if no space available.
                        *
D18B D5          SHREQ  PUSH  D
D18C 1600               MVI   D,:00     ) Required space in DE
D18E 5F                 MOV   E,A       )
D18F CDC5D1             CALL  :D1C5     Run Heap request
D192 23                 INX   H
D193 D1                 POP   D
D194 C9                 RET
                        *
                        *
                        *  ==============
                        *** HEAP HANDLER ***
                        *  ==============
                        *
                        *
                        ************************************
                        * INIT. HEAP SPACE TO ALL AVAILABLE *
                        ************************************
                        *
                        * The pointers for textbuffer and symboltable
                        * are updated correctly according to the requested
                        * Heapsize. The Heap is cleared: It starts with
                        * HSIZE-4 IOR #8000 and ends with #7FFF.
                        *
                        * Entry: DE: HEAP size (<32K).
                        * Exit:  BC preserved. AFDEHL corrupted.
                        *        Error if insufficient space.
                        *
D195 2A9F02      HINIT  LHLD  :029F     Start text buffer in HL
D198 D5                 PUSH  D
D199 E5                 PUSH  H
D19A D5                 PUSH  D
D19B EB                 XCHG           Start textbuf in DE
D19C 2A9B02             LHLD  :029B     Start Heap in HL
D19F EB                 XCHG
```



```
D1A0 CD1ADE             CALL  :DE1A     Calc current heapsize
D1A3 EB                 XCHG            Store it in DE
D1A4 E1                 POP   H         Get new HEAP size
D1A5 CD1ADE             CALL  :DE1A     Calculate difference
D1A8 D1                 POP   D         Get old start textbuf
D1A9 CDD1C9             CALL  :C9D1     Move textbuf and symtab
D1AC 229F02             SHLD  :029F     Update start textbuf
D1AF D1                 POP   D         Get new HEAP size
D1B0 2A9B02             LHLD  :029B     Get startaddr HEAP
D1B3 1B                 DCX   D
D1B4 1B                 DCX   D
D1B5 1B                 DCX   D
D1B6 1B                 DCX   D         HSIZE-4
D1B7 7A                 MOV   A,D
D1B8 F680               ORI   :80       Free space available
D1BA 77                 MOV   M,A       ) Set start Heap to
D1BB 23                 INX   H         ) HSIZE-4 IOR #8000
D1BC 73                 MOV   M,E       )
D1BD 23                 INX   H
D1BE 19                 DAD   D         Get end HEAP -2
D1BF 367F               MVI   M,:7F     ) Set it to
D1C1 23                 INX   H         ) #7FFF
D1C2 36FF               MVI   M,:FF     ) (end marker)
D1C4 C9                 RET
                        *
                        ****************
                        * HEAP REQUEST *
                        ****************
                        *
D1C5 CF          HREQU  RST   1          Run heap request
D1C6 0F                 DATA  :0F
D1C7 C9                 RET
                        *
D1C8 FF                 DATA  :FF
D1C9 FF                 DATA  :FF
D1CA FF                 DATA  :FF
D1CB FF                 DATA  :FF
D1CC FF                 DATA  :FF
D1CD FF                 DATA  :FF
D1CE FF                 DATA  :FF
D1CF FF                 DATA  :FF
D1D0 FF                 DATA  :FF
                        *
                        ************************
                        * part of EDIT (2EEC0) *
                        ************************
                        *
                        * Prints a text line.
                        *
D1D1 CAECEE      LD15   JZ    :EEEC      (2) If char=0
D1D4 F2DEEE             JP    :EEDE      (2) Cont for char #01-7F
D1D7 23                 INX   H          Next pos in textbuf
D1D8 C3D2EE             JMP   :EED2      (2) Ignore char >= #80
                        *
                        ******************************************
                        * INPUT SCANNING: CHECK SOURCE AND INPUTS *
                        ******************************************
                        *
                        * Part of D6BB.
                        * Checks if input from keyboard or from DINC.
                        * Checks also for any new inputs.
                        *
```

```
D1DB 3A9602     MPT29   LDA   :0296     Get input direction
D1DE B7                 ORA   A
D1DF C2E002             JNZ   :02E0     Jump if input from DINC

                * If input from keyboard:

D1E2 CDA5D6     IN0     CALL  :D6A5     Check if new keys pressed
D1E5 C3C1D6             JMP   :D6C1     Into keyb.scanning
                *
                ******************************
                * part of FPT COMPARE (C079) *
                ******************************
                *
                * Entry: Contents MACC in ABCD, exp.byte in E.
                *        HL points to exp.byte MEM.
                *
D1E8 78         LD17    MOV   A,B
D1E9 23                 INX   H
D1EA A6                 ANA   M         AND 1st bytes both mantissas
D1EB 7E                 MOV   A,M
D1EC 2B                 DCX   H         Pnts to exp.byte MEM
D1ED FAF5D1             JM    :D1F5     Jump if both 1st mantissa
                                        bits are '1'
D1F0 B8                 CMP   B         Comp both 1th mantissa bytes
D1F1 3F                 CMC             CY=0 if mantissa MACC >
                                        mantissa MEM
D1F2 C39FC0             JMP   :C09F     Quit
                *
D1F5 7B         LD18    MOV   A,E       Get exp.byte MACC
D1F6 AE                 XRA   M         XOR both exponents
D1F7 E640               ANI   :40       Check if only 1 exp. neg.
D1F9 7B                 MOV   A,E       Get exp.byte MACC
D1FA C387C0             JMP   :C087     Back to main routine
                *
                ************************
                * part of EDIT (2EE7B) *
                ************************
                *
                * Skip to B'th position on line.
                *
D1FD 7E         LD19    MOV   A,M       Get char in A
D1FE B7                 ORA   A         Set flags on it
D1FF FA94EE             JM    :EE94     (2) Ignore char >= #80
D202 78                 MOV   A,B       Get pos. required
D203 B9                 CMP   C         Reached ?
D204 7E                 MOV   A,M       Get char
D205 C382EE             JMP   :EE82     (2)
                *
                ********
                * DATA *
                ********
                *
D208 0D         MSG01   DATA  :0D       CR
D209 DB6F               DBL   :6FDB     'SOME INPUT IGNORED'
D20B 00                 DATA  :00
                *
                ***********************************
                * part of UNDERFLOW ERROR (C065) *
                ***********************************
                *
D20C E7         LD21    RST   4         Copy operand to MACC
D20D 0C                 DATA  :0C
```



```
D20E 210400             LXI   H,:0004
D211 C34FC0             JMP   :C04F     Cont on (00D0/1)+4
                *
                ****************************************
                * part of RUN 'CLEAR' (cont. of 0E6B5) *
                ****************************************
                *
D214 229D02     MPT44   SHLD  :029D     Update HEAP size
D217 2A0001             LHLD  :0100     Get start current line
D21A 7C                 MOV   A,H
D21B B5                 ORA   L         Check if CURRNT=0
D21C D1                 POP   D
D21D C8                 RZ              Abort if immediate cmd
D21E CD01E4             CALL  :E401     (0) Run RESTORE
D221 09                 DAD   B
D222 CD2DE9             CALL  :E92D     (0) Calc length of block
                                        Result in BC
D225 B7                 ORA   A
D226 C9                 RET
                *
                *********************************
                * part of HEAP REQUEST (3E9AD) *
                *********************************
                *
                * Checks if end of Heap reached.
                *
D227 FE7F       HR080   CPI   :7F       End marker ?
D229 C2FAE9             JNZ   :E9FA     (3) Jump if not
D22C 7B                 MOV   A,E       Get 2nd byte in A
D22D 3C                 INR   A
D22E C2FAE9             JNZ   :E9FA     (3) Jump if it was <> FF
D231 3E07               MVI   A,:07     If end of Heap reached:
D233 C3F5D9             JMP   :D9F5     Run error 'OUT OF STRING
                                        SPACE'
                *
                ************************
                * CLEAR A HEAP ENTRY *
                ************************
                *
                * A Heap entry is erased by setting the msb o
                * the first byte to 1.
                *
                * Entry: HL: Points to byte.
                * Exit:  All registers preserved.
                *
D236 F5         HREL    PUSH  PSW
D237 7E                 MOV   A,M       Get byte
D238 F680               ORI   :80       Set msb=1
D23A 77                 MOV   M,A       Store byte
D23B F1                 POP   PSW
D23C C9                 RET
                *
                *
                *
D23D                    END
```

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

HINIT  D195   HREL   D236   HREQU  D1C5   HR080  D227
IN0    D1E2   LD10   D177   LD11   D185   LD15   D1D1
```

LD17	D1E8	LD18	D1F5	LD19	D1FD	LD21	D20C
LD3	D128	LD4	D13F	LD5	D145	LD9	D175
MPT29	D1DB	MPT44	D214	MSG01	D208	SCOPF	D16D
SCOPT	D173	SHAPP	D106	SHCOMP	D121	SHCOPY	D172
SHINIT	D101	SHM05	D151	SHM08	D15A	SHMID	D14F
SHREL	D187	SHREQ	D18B				


```
002                             ORG   :D23D
003                     *
004                     *
005                     *
006                     *  =============
007                     *** I/O HANDLER ***
008                     *  =============
009                     *
010                     *
011                     **********************
012                     * RUN basiccmd SAVE *
013                     **********************
014                     *
015                     * Valid as direct command and in program.
016                     * Clears the Heap, zeroes all variables, evaluate
017                     * an evt. program name and writes a file of type
018                     * 0 (BASIC) on tape.
019                     *
020                     * Exit: HL: Points to end symboltable.
021                     *       DE: Length symboltable.
022                     *       BC: Updated.
023                     *
024 D23D CD23CB         RSAVE   CALL  :CB23     Empty HEAP + symtab
025 D240 2AA102                 LHLD  :02A1     Get start symtab
026 D243 EB                     XCHG            in DE
027 D244 2AA302                 LHLD  :02A3     End symtab in HL
028 D247 CD1ADE                 CALL  :DE1A     Calculate length symtab
029 D24A E5                     PUSH  H         Preserve length symtab
030 D24B 2A9F02                 LHLD  :029F     Get start textbuf
031 D24E EB                     XCHG            in DE
032 D24F CD1ADE                 CALL  :DE1A     Calculate length textbuf
033 D252 E5                     PUSH  H         Preserve length textbuf
034 D253 D5                     PUSH  D         Preserve start textbuf
035 D254 CD91E7                 CALL  :E791     (0) Evaluate program name
036 D257 00                     NOP
037 D258 00                     NOP
038 D259 00                     NOP
039 D25A 3E30                   MVI   A,:30     File type byte 0
040 D25C 00                     NOP
041 D25D 00                     NOP
042 D25E 00                     NOP
043 D25F 00                     NOP
044 D260 00                     NOP
045 D261 00                     NOP
046 D262 00                     NOP
047 D263 CDC502                 CALL  :02C5     Write fileleader, flagbyte,
048                                             file type byte, name length
049                                             and name.
050 D266 E1                     POP   H         Start textbuf in HL
051 D267 D1                     POP   D         Length textbuf in DE
052 D268 CDC802                 CALL  :02C8     Write length and contents
053                                             textbuf
054 D26B D1                     POP   D         Length symtab in DE
055 D26C C3D8D7                 JMP   :D7D8     Write length and contents
056                                             symtab + file trailer
057                     *
058 D26F FF                     DATA  :FF
059                     *
060                     **********************
061                     * RUN basiccmd LOAD *
062                     **********************
063                     *
```

```
064                     * Valid as direct command and in program.
065                     * Clears Heap and all variables. Evaluates name
066                     * of program, updates BC. Required file type: 0.
067                     * C is set to print type/name or not.
068                     * A file (type 0: Basic) is read from tape. When
069                     * a file has been found, the textbuffer and the
070                     * symboltable are loaded and the pointers updated.
071                     * When loading during program run: the program
072                     * continues with the program just loaded.
073                     *
074                     * Exit: No error: BC: Updated.
075                     *                 DE: Begin screen RAM.
076                     *                 HL: End symbol table.
077                     *
078 D270 CD23CB     RLOAD   CALL  :CB23     Empty HEAP + symtab
079 D273 CD91E7             CALL  :E791     (0) Evaluate programname
080 D276 C5                 PUSH  B
081 D277 00                 NOP
082 D278 00                 NOP
083 D279 00                 NOP
084 D27A 00                 NOP
085 D27B 0630               MVI   B,:30     File type byte 0
086 D27D E5                 PUSH  H         Preserve length name reqd
087 D27E 2A0001             LHLD  :0100     Get CURRNT
088 D281 7C                 MOV   A,H
089 D282 B5                 ORA   L         Load during run program?
090 D283 0E00               MVI   C,:00
091 D285 C289D2             JNZ   :D289     If during run; C=00
092 D288 0D                 DCR   C         Else C=FF
093 D289 E1         RLD10   POP   H
094 D28A CDCE02             CALL  :02CE     Switch on cassette motors;
095                                         read header + name
096 D28D 2AA502             LHLD  :02A5     Get end free RAM
097 D290 EB                 XCHG            Max. RAM in DE
098 D291 2A9F02             LHLD  :029F     Start textbuf in HL
099 D294 CDD102             CALL  :02D1     Load textbuffer
100 D297 22A102             SHLD  :02A1     Store end textbuffer
101 D29A DCD102             CC    :02D1     Load symboltable
102 D29D CD1AD7             CALL  :D71A     Store end symtab; stop
103                                         cassette motors.
104 D2A0 C1                 POP   B
105 D2A1 FB                 EI              Enable interrupts
106 D2A2 D2A8D2             JNC   :D2A8     If loading error
107 D2A5 3E00               MVI   A,:00     No loading error
108 D2A7 C9                 RET
109                     *
110                     **********************
111                     * RUN LOADING ERROR *
112                     **********************
113                     *
114                     * The programbuffers are restored. A error message
115                     * is printed.
116                     *
117 D2A8 F5         RLERR   PUSH  PSW
118 D2A9 CDB5DE             CALL  :DEB5     Run 'NEW'
119 D2AC F1                 POP   PSW
120 D2AD 210000             LXI   H,:0000
121 D2B0 220001             SHLD  :0100     Set CURRNT=0
122 D2B3 C60B       RLEAR   ADI   :0B
123 D2B5 C3F5D9             JMP   :D9F5     Run 'LOADING ERROR ..'
124                     *
125                     *
```



```
126                     ******************
127                     * OPEN TAPE FILE *
128                     ******************
129                     *
130                     * Entry: A:  File type.
131                     *        HL: Points to file name.
132                     * Exit:  HL: Points beyond file name.
133                     *        DE: Length of name.
134                     *        BC: Preserved.
135                     *        A:  Checksum on name.
136                     *
137 D2B8 F5         CWOPEN  PUSH  PSW
138 D2B9 CD20D7             CALL  :D720     Init. write file leader
139 D2BC F1                 POP   PSW
140 D2BD CD09D5             CALL  :D509     Write file type byte
141 D2C0 C3F8D7             JMP   :D7F8     Get name length, write it
142                                         on tape, incl. its c.s.
143                     *
144                     **********************
145                     * RUN basiccmd CHECK *
146                     **********************
147                     *
148                     * Valid as direct command only.
149                     * Checks on file type and name. For all files
150                     * with type <3, a checksum on all data is done.
151                     * This routine remains in a endless loop and
152                     * can be aborted with BREAK only.
153                     *
154                     RCHECK
155 D2C3 210000       CHK10   LXI   H,:0000   No program name given
156 D2C6 01FF00             LXI   B,:00FF   Any file type
157 D2C9 00                 NOP
158 D2CA 00                 NOP
159 D2CB CDCE02             CALL  :02CE     Read file header, file
160                                         type and name; print
161                                         type and name
162 D2CE 00                 NOP
163 D2CF FE33               CPI   :33
164 D2D1 D2EBD2             JNC   :D2EB     If file type >=3: no check
165                                         on checksum
166
167                     * Test checksum::
168
169 D2D4 0C                   INR   C         BC=0
170 D2D5 CDE6D7             CALL  :D7E6     Set A=0, read + check
171                                         a data block
172 D2D8 CCD702             CZ    :02D7     Read + check next block
173 D2DB C2E6D2             JNZ   :D2E6     If reading error
174 D2DE CDFFDA             CALL  :DAFF     Print 'OK', car.ret
175 D2E1 C0DB               DBL   :DBC0
176 D2E3 C3C3D2             JMP   :D2C3     Wait for next file
177
178                     * If checksum error:
179
180 D2E6 CDFFDA       CHK20   CALL  :DAFF     Print 'BAD'
181 D2E9 DBDB               DBL   :DBDB
182 D2EB CD5EDD       CHK30   CALL  :DD5E     Print car.ret
183 D2EE C3C3D2             JMP   :D2C3     Wait for next file
184                     *
185                     **********************
186                     * WRITE BLOCK ON TAPE *
187                     **********************
```

Ropen D3-25

```
                        *
                        * 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
                        *
```


Ropen.

```
                        **********************
                        * 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
```

```
312 D356 3C              INR   A         Loading error 1
313 D357 E1              POP   H
314 D358 E5              PUSH  H         Restore begin addr.
315 D359 DA80D3          JC    :D380     If loading error 1
316 D35C 0656            MVI   B,:56     Init checksum
317 D35E 7A        LBK10 MOV   A,D
318 D35F B3              ORA   E
319 D360 CA6FD3          JZ    :D36F     If whole block read
320 D363 1B              DCX   D
321 D364 CD84D3          CALL  :D384     Read next byte, update c.s
322 D367 DA7ED3          JC    :D37E     If loading error 3
323 D36A 77              MOV   M,A       Store byte in buffer
324 D36B 23              INX   H
325 D36C C35ED3          JMP   :D35E     Next byte
326
327                * If whole block read:
328
329 D36F CDD4D4    LBK20 CALL  :D4D4     Read checksum block contents
330 D372 DA7ED3          JC    :D37E     If loading error 3
331 D375 B8              CMP   B         Check checksum
332 D376 3E02            MVI   A,:02     Loading error 2
333 D378 C280D3          JNZ   :D380     If loading error 2
334 D37B C3B4C6          JMP   :C6B4     CY=1, return: no error
335
336                * If loading error:
337
338 D37E 3E03      LBK40 MVI   A,:03     Loading error 3
339 D380 B7        LOERR ORA   A
340 D381 C3B6C6          JMP   :C6B6     Return with CY=0: error
341                *
342                **********************************
343                * READ BYTE, CALCULATE CHECKSUM *
344                **********************************
345                *
346                * Entry: B: Checksum.
347                * Exit:  A: Byte read.
348                *        B: Updated checksum.
349                *        CDEHL preserved.
350                *
351 D384 CDD4D4    INSC  CALL  :D4D4     Read byte
352 D387 F5        RBUEX PUSH  PSW
353 D388 A8              XRA   B         Calculate checksum
354 D389 07              RLC
355 D38A 47              MOV   B,A       Store new value
356 D38B F1              POP   PSW
357 D38C C9              RET
358                *
359                ********************
360                * READ NAME LENGTH *
361                ********************
362                *
363                * Entry: No conditions.
364                * Exit:  HL: Length name read.
365                *        A:  Result checksum check (0 if OK).
366                *        BCDE: preserved.
367                *        CY=0: O.K.; CY=1: Out of data.
368                *
369 D38D C5        INLNG PUSH  B
370 D38E 0656            MVI   B,:56     Init. checksum
371 D390 CD84D3          CALL  :D384     Read highest length byte
372                                       and update checksum
373 D393 67              MOV   H,A
```



```
374 D394 D484D3          CNC   :D384     Read lowest length byte
375                                       and update checksum
376 D397 6F              MOV   L,A
377 D398 D4D4D4          CNC   :D4D4     Read checksum on length
378 D39B F5        RHLEX PUSH  PSW
379 D39C 90              SUB   B         Check checksum
380 D39D 47              MOV   B,A
381 D39E F1              POP   PSW
382 D39F 78              MOV   A,B
383 D3A0 C1              POP   B
384 D3A1 C9              RET
385                *
386                ******************************************
387                * READ + CHECK PROGRAM NAME AND FILE TYPE *
388                ******************************************
389                *
390                * Routine searches for proper file name by
391                * reading file name and compare it with name
392                * requested.
393                *
394                * Entry: A:  Evt. difference in file type byte
395                *            read and requested.
396                *        B:  Requested file type.
397                *        C:  00 during run program, else FF.
398                *        DE: Length requested.
399                *        HL: Address 1st byte name requested.
400                * Exit:  BCDEHL preserved.
401                *        A=0: All OK.
402                *        A=1: Loading error 1.
403                *
404 D3A2 C5        CMBLK PUSH  B         Save file type + RUN flag
405 D3A3 E5              PUSH  H         Save addr reqd name
406 D3A4 47              MOV   B,A       Store deviation file type
407 D3A5 D5        MBK10 PUSH  D         Save req. name length
408 D3A6 E5              PUSH  H
409 D3A7 CD8DD3          CALL  :D38D     Read + check program name
410                                       evt. c.s.failure in A
411 D3AA DAEDD3          JC    :D3ED     If reading error
412 D3AD B7              ORA   A
413 D3AE C2EDD3          JNZ   :D3ED     If checksum error
414 D3B1 E5              PUSH  H         Save length name on tape
415 D3B2 CD1ADE          CALL  :DE1A     Calculate difference name
416                                       lengths reqd and on tape
417 D3B5 7C              MOV   A,H
418 D3B6 B5              ORA   L
419 D3B7 67              MOV   H,A       Difference in H
420 D3B8 68              MOV   L,B       Difference file type in L
421 D3B9 D1              POP   D         Get length name on tape
422 D3BA 0656            MVI   B,:56     Initiate checksum
423 D3BC E3        MBK20 XTHL            Get byte reqd name
424 D3BD 7A              MOV   A,D
425 D3BE B3              ORA   E         Length name on tape = 0 ?
426 D3BF CAD8D3          JZ    :D3D8     If length = 0, or whole
427                                       name read.
428 D3C2 1B              DCX   D
429 D3C3 CD84D3          CALL  :D384     Read bytes of name, update
430                                       checksum
431 D3C6 DAEDD3          JC    :D3ED     If reading error
432 D3C9 0D              DCR   C
433 D3CA 0C              INR   C         Load during run?
434 D3CB F5              PUSH  PSW       Save length name on tape
435 D3CC C4EBD7          CNZ   :D7EB     Display program name
```

```
436 D3CF F1                 POP   PSW       Get byte of name on tape
437 D3D0 AE                 XRA   M         Compare with name reqd
438 D3D1 23                 INX   H
439 D3D2 E3                 XTHL            Get 'difference flag'
440 D3D3 B4                 ORA   H         Update it
441 D3D4 67                 MOV   H,A       and store it in H
442 D3D5 C3BCD3             JMP   :D3BC     Next byte
443
444                 * If whole name read:
445
446 D3D8 CDD4D4   MBK30     CALL  :D4D4     Read c.s on name contents
447 D3DB DAEDD3             JC    :D3ED     If reading error
448 D3DE A8       MBEX      XRA   B         Check checksum
449 D3DF E1                 POP   H
450 D3E0 B5                 ORA   L         Check file type
451 D3E1 6F                 MOV   L,A
452 D3E2 D1                 POP   D         Get length req. name
453 D3E3 7A                 MOV   A,D
454 D3E4 B3                 ORA   E         No name requested ?
455 D3E5 CAE9D3             JZ    :D3E9     If load without name
456 D3E8 7C                 MOV   A,H       Difference in names?
457 D3E9 B5       MBK40     ORA   L         Take also other checks in
458                                           account
459 D3EA E1       MBK45     POP   H
460 D3EB C1                 POP   B
461 D3EC C9                 RET
462
463                 * If error:
464
465 D3ED E1        MBK50     POP   H
466 D3EE D1                 POP   D
467 D3EF 3E01               MVI   A,:01     Loading error 1
468 D3F1 C3E9D3             JMP   :D3E9
469                 *
470                 *
471                 *
472 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   D289    RLEAR   D2B3    RLERR   D2A8    RLOAD   D270
RPN10   D329    RSAVE   D23D
```



```
002                         ORG   :D3F4
003                 *
004                 *
005                 *
006                 **********************
007                 * READ FILE HEADER *
008                 **********************
009                 *
010                 * Locates a file on tape and reads leader.
011                 * Exit: Interrupts are disabled. BCDEHL preserved.
012                 *
013 D3F4 CD8FD9     RHDR      CALL  :D98F     Disable sound interrupt
014 D3F7 CD80D4               CALL  :D480     Find sync pattern
015 D3FA CDD4D4               CALL  :D4D4     Read flag type byte
016 D3FD DAF4D3               JC    :D3F4     Again if reading error
017 D400 FE55                 CPI   :55
018 D402 C2F4D3               JNZ   :D3F4     Again if not flag byte
019 D405 CDD4D4               CALL  :D4D4     Read file type byte
020 D408 DAF4D3               JC    :D3F4     Again if reading error
021 D40B C9                   RET
022                 *
023                 **********************
024                 * WRITE FILE LEADER *
025                 **********************
026                 *
027                 * Writes a leader for program or data block on tape.
028                 * Disables interrupts which could cause problems.
029                 *
030                 * Entry: at WHDR:  Entry if not during run program.
031                 *        at WHD20: If during run of program.
032                 * Exit:  BCDEHL preserved.
033                 *
034 D40C CDFFDA     WHDR      CALL  :DAFF     Print 'SET RECORD, START
035 D40F 9CDB                 DBL   :DB9C     TAPE, TYPE SPACE'
036 D411 CDCBD7               CALL  :D7CB     Wait for spacebar pressed
037 D414 CD2ED4     WHD20     CALL  :D42E     Switch on cassette motors
038 D417 F3                   DI              Disable interrupts
039 D418 00                   NOP
040 D419 00                   NOP
041 D41A CDEDD4               CALL  :D4ED     Write leader
042 D41D 3E55                 MVI   A,:55     Get flag byte
043 D41F C309D5               JMP   :D509     Write flag byte
044                 *
045                 **************
046                 * (Not used) *
047                 **************
048                 *
049 D422 CDF1D2     MPT27     CALL  :D2F1     Write block on tape
050 D425 00                   NOP
051 D426 00                   NOP
052                 *
053                 **********************
054                 * WRITE FILE TRAILER *
055                 **********************
056                 *
057                 * Write a trailer for program or datablock.
058                 *
059                 * Entry: Length of trailer in C.
060                 * Exit:  A=0, BCDEHL preserved.
061                 *
062                 WTRL
063 D427 CD50D5     CWCLOS    CALL  :D550     Write trailer bytes
```

```
064 D42A FB              EI              Enable interrupts
065 D42B C345D4          JMP   :D445     Stop cassette motors
066                  *
067                  ************************
068                  * START CASSETTE MOTORS *
069                  ************************
070                  *
071                  * Turns on motor of selected cassettedeck and
072                  * waits 665 msec.
073                  *
074                  * Exit: All registers preserved.
075                  *
076 D42E F5          CASST   PUSH  PSW
077 D42F 3A4000              LDA   :0040     Load POROM
078 D432 F630                ORI   :30       Disable cassette motors
079 D434 E5                  PUSH  H
080 D435 213D01              LXI   H,:013D   Addr CASSL
081 D438 AE                  XRA   M         Get selected cassette
082 D439 E1                  POP   H
083 D43A 324000              STA   :0040     Remember POROM
084 D43D 3206FD              STA   :FD06     Switch cassette motor on
085 D440 CD41DE              CALL  :DE41     Delay
086 D443 F1                  POP   PSW
087 D444 C9                  RET
088                  *
089                  ************************
090                  * STOP CASSETTE MOTORS *
091                  ************************
092                  *
093                  * Switches off cassettemotors.
094                  *
095                  * Exit: All registers preserved.
096                  *
097                  CRCLOS
098 D445 F5          CASSP   PUSH  PSW
099 D446 3A4000              LDA   :0040     Load POROM
100 D449 F630                ORI   :30       Disable cassette motors
101 D44B 324000              STA   :0040     Remember POROM
102 D44E 3206FD              STA   :FD06     Switch cassette motors off
103 D451 F1                  POP   PSW
104 D452 C9                  RET
105                  *
106                  ************
107                  * READ BIT *
108                  ************
109                  *
110                  * Reads one bit from tape.
111                  *
112                  * Entry: Address input port in HL. Input low state.
113                  * Exit:  CY=0: sign bit of A is bit read.
114                  *        CY=1: reading error.
115                  *        EHL preserved.
116                  *
117 D453 AF          RBIT    XRA   A
118 D454 57                  MOV   D,A
119 D455 47                  MOV   B,A
120 D456 4F                  MOV   C,A
121
122                  * 1st impulse:
123
124 D457 05          RBT10   DCR   B
125 D458 CA7ED4              JZ    :D47E     Too long low
```



```
126 D45B B6                  ORA   M
127 D45C F257D4              JP    :D457     Wait for high
128 D45F 0D          RBT30   DCR   C
129 D460 CA7ED4              JZ    :D47E     Too long high
130 D463 15                  DCR   D
131 D464 A6                  ANA   M
132 D465 FA5FD4              JM    :D45F     Wait low
133 D468 010000              LXI   B,:0000
134
135                  * 2nd impulse:
136
137 D46B 05          RBT40   DCR   B
138 D46C CA7ED4              JZ    :D47E     Too long low
139 D46F B6                  ORA   M
140 D470 F26BD4              JP    :D46B     Wait high again
141 D473 0D          RBT50   DCR   C
142 D474 CA7ED4              JZ    :D47E     Too long high
143 D477 14                  INR   D
144 D478 A6                  ANA   M
145 D479 FA73D4              JM    :D473     Wait low
146 D47C 7A                  MOV   A,D
147 D47D C9                  RET
148
149                  * If error:
150
151 D47E 37          RBT90   STC             Set CY if error
152 D47F C9                  RET
153                  *
154                  ***************
155                  * READ LEADER *
156                  ***************
157                  *
158                  * Finds a section of leader on the tape.
159                  *
160                  * Entry: No conditions.
161                  * Exit:  BCDEHL preserved, interrupts disabled.
162                  *
163 D480 C5          RLEAD   PUSH  B
164 D481 D5                  PUSH  D
165 D482 E5                  PUSH  H
166 D483 0628                MVI   B,:28     Estimate of impulse length
167 D485 2100FD              LXI   H,:FD00   address input port
168 D488 3EFF        RDL05   MVI   A,:FF
169 D48A FB          RDL10   EI              Enable interrupts
170 D48B 00                  NOP             (here cursor flashes)
171 D48C F3                  DI              Disable interrupts
172 D48D A6                  ANA   M
173 D48E FA8AD4              JM    :D48A     Wait low
174 D491 48                  MOV   C,B       Estimated length in C
175 D492 1614                MVI   D,:14     Needs this many cycles for
176                                          synchronisation. Must be
177                                          more than trailer length.
178 D494 1E00        RDL30   MVI   E,:00
179 D496 AF                  XRA   A
180 D497 1D          RDL40   DCR   E
181 D498 CA88D4              JZ    :D488     Too long low; start again
182 D49B B6                  ORA   M
183 D49C F297D4              JP    :D497     Wait high
184 D49F 0600                MVI   B,:00
185 D4A1 04          RDL50   INR   B
186 D4A2 CA88D4              JZ    :D488     Too long high; start again
187 D4A5 A6                  ANA   M
```

```
188 D4A6 FAA1D4            JM    :D4A1     Wait low
189 D4A9 78                MOV   A,B
190 D4AA 91                SUB   C         Compare impulse length
191                                        with estimate
192 D4AB F2B0D4            JP    :D4B0
193 D4AE 2F                CMA             )
194 D4AF 3C                INR   A         ) 2-complement if <0
195 D4B0 5F          RDL60 MOV   E,A       Store difference in E
196 D4B1 79                MOV   A,C       Calculate margin
197 D4B2 E6F0              ANI   :F0
198 D4B4 1F                RAR
199 D4B5 1F                RAR
200 D4B6 1F                RAR             Margin: 1/8th of estimate
201 D4B7 BB                CMP   E         Compare with difference
202 D4B8 DAC3D4            JC    :D4C3     Not within margin
203
204                  * If sync archieved:
205
206 D4BB 15                DCR   D
207 D4BC C294D4            JNZ   :D494     Next impulse until D=0
208 D4BF 14                INR   D
209 D4C0 C394D4            JMP   :D494     Next impulse until out
210                                        of margin
211
212                  * If out of margin:
213
214 D4C3 15          RDL70 DCR   D
215 D4C4 C288D4            JNZ   :D488     Not synchronised; again
216 D4C7 AF                XRA   A
217 D4C8 B6          RDL80 ORA   M
218 D4C9 F2C8D4            JP    :D4C8     Wait high
219 D4CC A6          RDL90 ANA   M
220 D4CD FACCD4            JM    :D4CC     Wait low
221 D4D0 E1                POP   H
222 D4D1 D1                POP   D
223 D4D2 C1                POP   B
224 D4D3 C9                RET
225                  *
226                  *************
227                  * READ BYTE *
228                  *************
229                  *
230                  * Reads one byte from tape.
231                  *
232                  * Entry: No conditions.
233                  * Exit:  CY=0: Byte read in A.
234                  *        CY=1: Some error.
235                  *        BCDEHL preserved.
236                  *
237 D4D4 C5          RBYTE PUSH  B
238 D4D5 D5                PUSH  D
239 D4D6 E5                PUSH  H
240 D4D7 2100FD            LXI   H,:FD00   Address input port
241 D4DA 1EFE              MVI   E,:FE
242 D4DC CD53D4      RBY10 CALL  :D453     Read bit
243 D4DF DAE9D4            JC    :D4E9     If reading error: CY=1
244 D4E2 17                RAL
245 D4E3 7B                MOV   A,E
246 D4E4 17                RAL
247 D4E5 5F                MOV   E,A       Shift bit into E
248 D4E6 DADCD4            JC    :D4DC     Next bit
249 D4E9 E1          RBY20 POP   H         8 bits read, no error
```



```
250 D4EA D1                POP   D
251 D4EB C1                POP   B
252 D4EC C9                RET
253                  *
254                  ****************
255                  * WRITE LEADER *
256                  ****************
257                  *
258                  * Writes a leader on the tape. From WLD10 also used
259                  * to write a trailer.
260                  *
261                  * Entry: No conditions.
262                  * Exit:  A=0, BCDEHL preserved.
263                  *
264 D4ED 00          WLEAD NOP
265 D4EE C5                PUSH  B
266 D4EF E5                PUSH  H
267 D4F0 2AE602            LHLD  :02E6     Get leader impulse length
268 D4F3 01E807            LXI   B,:07E8   Period for synchr.
269 D4F6 CD24D5      WLD10 CALL  :D524     Write bit
270 D4F9 0B                DCX   B
271 D4FA 78                MOV   A,B
272 D4FB B1                ORA   C
273 D4FC C2F6D4            JNZ   :D4F6     Write many bits
274 D4FF 2AE802            LHLD  :02E8     Get impulse length data bit
275 D502 CD24D5            CALL  :D524     Write a data '1' bit to end
276 D505 E1                POP   H
277 D506 C1                POP   B
278 D507 00                NOP
279 D508 C9                RET
280                  *
281                  **************
282                  * WRITE BYTE *
283                  **************
284                  *
285                  * Write a byte to tape.
286                  *
287                  * Entry: Byte to be written in A.
288                  * Exit:  All registers preserved.
289                  *
290 D509 F5          WBYTE PUSH  PSW
291 D50A C5                PUSH  B
292 D50B D5                PUSH  D
293 D50C E5                PUSH  H
294 D50D 2AE802            LHLD  :02E8     Get impulse length bit '1'
295 D510 5C                MOV   E,H
296 D511 55                MOV   D,L       DE: inpulse length bit '0'
297 D512 0608              MVI   B,:08     8 bits to write
298 D514 17          WBY10 RAL             Set/reset CY for kind of bit
299 D515 DC24D5            CC    :D524     Write data '1' bit
300 D518 EB                XCHG
301 D519 D424D5            CNC   :D524     Write data '0' bit
302 D51C EB                XCHG
303 D51D 05                DCR   B
304 D51E C214D5            JNZ   :D514     Next bit
305 D521 C356CB            JMP   :CB56     Pop all, ret
306                  *
307                  *************
308                  * WRITE BIT *
309                  *************
310                  *
311                  * 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  D4B0   WBIT   D524   WBY10  D514
WBYTE  D509   WCY10  D542   WCY20  D54B   WCYC   D53C
WHD20  D414   WHDR   D40C   WLD10  D4F6   WLEAD  D4ED
WTRL   D427   WTRLX  D550
```

```
002                     ORG   :D620
003                   *
004                   *
005                   *
006                   ***************************
007                   * COMPLETE KEYBOARD SCAN *
008                   ***************************
009                   *
010                   * Initialises a complete keyboard scan,
011                   * independent of the KNSCAN flag, and performs it.
012                   *
013                   * Exit: All registers preserved.
014                   *
015 D620 F5           KFSCAN  PUSH  PSW
016 D621 C5                   PUSH  B
017 D622 D5                   PUSH  D
018 D623 E5                   PUSH  H
019 D624 21B902               LXI   H,:02B9   Addr KNSCAN pointer
020 D627 3600                 MVI   M,:00     Enable complete scan
021 D629 E5                   PUSH  H
022 D62A CD9AD5               CALL  :D59A     Scan keyboard and store
023                                           result in circ.buffer
024 D62D E1                   POP   H
025 D62E 35                   DCR   M         Scan for BREAK only
026 D62F C356CB               JMP   :CB56     Popall; return
027                   *
028                   ***********************************
029                   * GET ASCII VALUE OF KEY PRESSED *
030                   ***********************************
031                   *
032                   * Calculates address in ASCII table in ROM and gets
033                   * ASCII value of the key pressed. The result is
034                   * stored in the 4-bye circular buffer KLIND.
035                   *
036                   * Entry: A:  Keycode of scanned row (7 bits only).
037                   *        B:  FF when REPT pressed; else 00.
038                   *        C:  Number of row.
039                   *        DE: Address in MAP2.
040                   *        HL: Address in MAP1.
041                   * Exit:  BCDEHL preserved; AF corrupted.
042                   *
043 D632 C5           TKEY    PUSH  B
044 D633 0607                 MVI   B,:07     Check which key in row is
045 D635 1F           TKY10   RAR             pressed; calculate offset
046 D636 DC3FD6               CC    :D63F     Get ASCII value; store it
047                                           in KLIND
048 D639 05                   DCR   B
049 D63A C235D6               JNZ   :D635     Next column
050 D63D C1                   POP   B
051 D63E C9                   RET
052                   *
053                   * GET KEY-ASCII VALUE AND STORE IT:
054                   *
055 D63F CF           SINKEY  RST   1         Get ASCII-value from ROM-
056 D640 12                   DATA  :12       table and store it in KLIND
057 D641 C9                   RET
058                   *
059                   *********************************
060                   * OUTPUT TO RS232 IF REQUIRED *
061                   *********************************
062                   *
063                   * Checks if output is to RS232. If positive,
```



```
064                   * output is performed.
065                   *
066                   * Entry: Byte to be transmitted in A.
067                   *
068 D642 F5           TOUTSE  PUSH  PSW
069 D643 3A3101               LDA   :0131     Get output direction
070 D646 B7                   ORA   A         Check if RS232 output
071 D647 C28CDD               JNZ   :DD8C     Abort if not
072 D64A F1                   POP   PSW
073 D64B C394DD               JMP   :DD94     Output to RS232
074                   *
075                   *****************************************
076                   * GET ASCII-VALUE OF CHARACTER IN BUFFER *
077                   *****************************************
078                   *
079                   * Routine is not used.
080                   *
081                   * The Ascii-value of a character is stored in
082                   * KLIND. Afterwards, Bank select is restored.
083                   *
084 D64E CD3FE9        LD67    CALL  :E93F     (3) Ascii-value in KLIND
085 D651 F1                   POP   PSW       A contains POROM
086 D652 CD08D8               CALL  :D808     Update PORO and POROM
087 D655 F1                   POP   PSW
088 D656 E1                   POP   H
089 D657 C9                   RET
090                   *
091                   ************************************
092                   * parts of RUN 'RANDOMISE' (0E40C) *
093                   ************************************
094                   *
095                   RMI15
096 D658 3D           MPT39   DCR   A
097 D659 C258D6               JNZ   :D658     Again till A=0
098 D65C 7E                   MOV   A,M       Get contents FD00
099 D65D AB                   XRA   E
100 D65E C9                   RET
101                   *
102                   * Entry: L = 0.
103                   *
104 D65F 3AC101       MPT38   LDA   :01C1     Get keyb.scan time count
105                                           (0, 1 or 2)
106 D662 0F                   RRC
107 D663 0F                   RRC             A= 0, #40 or #80
108 D664 5F                   MOV   E,A       in E
109 D665 45                   MOV   B,L       )
110 D666 4D                   MOV   C,L       ) BC=0
111 D667 C9                   RET
112                   *
113                   ************************************
114                   * WRITE 2 BLOCKS + TRAILER ON TAPE *
115                   ************************************
116                   *
117                   * Entry: HL:    Startaddress 1st block.
118                   *        Stack: Length 2nd block.
119                   *               Startaddress 2nd block.
120                   *
121 D668 110100       MPT13   LXI   D,:0001   Length 1st block = 1
122 D66B CDC802               CALL  :02C8     Write 1st block
123 D66E D1                   POP   D         Get length 2nd block
124 D66F E1                   POP   H         Get startaddr. 2nd block
125 D670 CDC802               CALL  :02C8     Write 2nd block
```

```
126 D673 CDCB02             CALL  :02CB     Write trailer
127 D676 B7                 ORA   A
128 D677 C9                 RET
129                 *
130                 *********************************
131                 * LOADA: EVALUATE PROGRAM NAME *
132                 *********************************
133                 *
134                 * The program name is evaluated. Selection of
135                 * ROM bank 1 is prepared.
136                 *
137 D678 CD87D6      MPT14   CALL  :D687     Evaluate program name
138 D67B 3A4000             LDA   :0040     Get POROM
139 D67E F640               ORI   :40       Prepare selection ROMbank 1
140 D680 C9                 RET
141                 *
142                 ******************
143                 * OPEN READ FILE *
144                 ******************
145                 *
146 D681 D5         MPT18   PUSH  D
147 D682 CDCE02             CALL  :02CE     Open READ file
148 D685 D1                 POP   D
149 D686 C9                 RET
150                 *
151                 *********************************
152                 * EVALUATE A STRING EXPRESSSION *
153                 *********************************
154                 *
155                 * A string expression is evaluated. Eventually,
156                 * the Heap entry is cleared if the string was
157                 * temporarily on Heap.
158                 *
159                 * Exit: DE preserved, BC updated.
160                 *       HL points after string
161                 *
162 D687 D5         MPT15   PUSH  D
163 D688 CD91E7               CALL  :E791     (0) Eval. string expr.
164                                             evt. clear Heap entry
165 D68B D1                 POP   D
166 D68C C9                 RET
167                 *
168                 ********************
169                 * CURSOR HANDLING *
170                 ********************
171                 *
172                 * Load the cursor pointers with the address, the
173                 * colour and the contents of a new cursor address.
174                 *
175                 * Entry: HL: New cursor address.
176                 *         D:  The colour byte of this location.
177                 *         E:  The contents of this location.
178                 * Exit:  HL and DE exchanged; ABCF preserved.
179                 *
180 D68D 227200      SPT00   SHLD  :0072     Store cursor address
181 D690 EB                  XCHG
182 D691 227600              SHLD  :0076     Store cursor addr.contents
183 D694 C9                 RET
184                 *
185                 ************************
186                 * OUTPUT ONE CHARACTER *
187                 ************************
```



```
188                 *
189                 * Output direction depending on OTSW.
190                 * This routine is useable for all data output
191                 * functions in machine language programs.
192                 *
193                 * Entry: Character for output in A.
194                 * Exit:  All registers preserved.
195                 *
196 D695 F5          MPT31   PUSH  PSW
197 D696 CD60DD               CALL  :DD60     Output character in A.
198 D699 F1                  POP   PSW
199 D69A C9                 RET
200                 *
201 D69B FF                 DATA  :FF
202                 *
203                 ************************************************
204                 * KEYB. SCANNING: UPDATE POINTER OUTPUT BUFFER *
205                 ************************************************
206                 *
207                 * Updates the pointer to the circular output
208                 * buffer #02BA-#02BD.
209                 *
210                 * Entry: HL: KLIOU.
211                 * Exit:  HL: Updated KLIOU.
212                 *        AF corrupted. BCDE preserved.
213                 *
214 D69C 23          KPTRU   INX   H         Incr. KLIOU
215 D69D 7D                  MOV   A,L       Lobyte into A
216 D69E FEBE                CPI   :BE       Buffer full?
217 D6A0 C0                  RNZ             Quit if not
218 D6A1 21BA02              LXI   H,:02BA   Else: wrap around
219 D6A4 C9                  RET
220                 *
221                 ******************************************
222                 * KEYBOARD SCANNING: CHECK IF NEW INPUTS *
223                 ******************************************
224                 *
225                 * Returns a flag if BREAK has been pressed or if
226                 * there is a character available.
227                 *
228                 * Entry: No conditions.
229                 * Exit:  BCDEHL preserved.
230                 *        A: Difference KLIIN and KLIOU.
231                 *        CY=1: Break pressed.
232                 *        CY=0, Z=1: No inputs.
233                 *        CY=0, Z=0: New input available.
234                 *
235                 ASKKEY
236 D6A5 E5          BREAK   PUSH  H
237
238                 * If suspended:
239
240 D6A6 21C402              LXI   H,:02C4   Addr break pntr
241 D6A9 7E                  MOV   A,M
242 D6AA 3D                  DCR   A
243 D6AB FEFE                CPI   :FE       Test if not 0 or FF
244 D6AD DAB9D6              JC    :D6B9     Abort if break: CY=1
245                 *
246 D6B0 2AC002              LHLD  :02C0     Get KLIOU
247 D6B3 3ABE02              LDA   :02BE     Get KLIIN
248 D6B6 95                  SUB   L         New keys pressed?
249 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 *
                        ************************
```

```
498                     *
499 D783 05             MPT54   DCR   B
500 D784 04                     INR   B
501 D785 C2DED7                 JNZ   :D7DE     If file type byte <>0:
502                                             Run error
503 D788 F1                     POP   PSW
504 D789 C9                     RET
505                     *
506                     ************************************
507                     * CHECK IF LOAD DURING RUN PROGRAM *
508                     ************************************
509                     *
510                     * Entry: C: 00 if load during run, else it is FF.
511                     *         A: File type byte.
512                     * Exit:  ABCDEHL preserved.
513                     *
514 D78A 0D             MPT24   DCR   C
515 D78B 0C                     INR   C         Check C
516 D78C C8                     RZ              Abort if during run
517 D78D C3EBD7                 JMP   :D7EB     Display file type byte
518                     *
519                     *
520                     *
521 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
```



```
002                             ORG   :D790
003                     *
004                     *
005                     *
006                     ************************
007                     * CHECK FREE RAM SPACE *
008                     ************************
009                     *
010                     * Entry: DE: Startaddress.
011                     *        HL: 1st not useable address.
012                     * Exit:  HL: Useable RAM space.
013                     *        ABCDE preserved.
014                     *
015 D790 CD1ADE         MPT25   CALL  :DE1A     Calculate free space
016 D793 2B                     DCX   H
017 D794 C9                     RET
018                     *
019                     ***********************************
020                     * TRANSFER DATA/CASSETTE VECTORS *
021                     ***********************************
022                     *
023                     * Transfer data/cassette switching vectors from
024                     * ROM to RAM vector area.
025                     *
026                     * Exit: AFBC preserved. DEHL corrupted.
027                     *
028                     MPT01
029 D795 C5             CASIN   PUSH  B
030 D796 21CBD7                 LXI   H,:D7CB   Highest source address
031 D799 11A4D7                 LXI   D,:D7A4   Lowest source address
032 D79C 01C502                 LXI   B,:02C5   Lowest destination address
033 D79F CD4FDE                 CALL  :DE4F     Transfer cassette vectors
034 D7A2 C1                     POP   B
035 D7A3 C9                     RET
036                     *
037                     ************************************
038                     * DATA/CASSETTE SWITCHING VECTORS *
039                     ************************************
040                     *
041                     * This data block is moved into the RAM area
042                     * #02C5-#02EB during system initialisation.
043                     *
044 D7A4 C3B8D2         CINTB   JMP   :D2B8     WOPEN
045 D7A7 C3F1D2                 JMP   :D2F1     WBLK
046 D7AA C327D4                 JMP   :D427     WCLOSE
047 D7AD C325D3                 JMP   :D325     ROPEN
048 D7B0 C340D3                 JMP   :D340     RBLK
049 D7B3 C345D4                 JMP   :D445     RCLOSE
050 D7B6 C3A2D3                 JMP   :D3A2     MBLK
051 D7B9 C9                     RET             RESET
052 D7BA 00                     NOP
053 D7BB 00                     NOP
054 D7BC C9                     RET             DOUTC
055 D7BD 00                     NOP
056 D7BE 00                     NOP
057 D7BF C3B4DD                 JMP   :DDB4     DINC
058 D7C2 C9                     RET
059 D7C3 00                     NOP
060 D7C4 00                     NOP
061 D7C5 2424                   DATA  :24,:24   TAPSL
062 D7C7 243C                   DATA  :24,:3C   TAPSD
063 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           )
```

```
188 D818 C1                 POP   B          )
189 D819 C3EFEA             JMP   :EAEF      (2)
190                 *
191 D81C FF                 DATA  :FF
192                 *
193                 **********************
194                 * RUN basiccmd SAVEA *
195                 **********************
196                 *
197 D81D CD3BD8     RSAVA   CALL  :D83B      Evaluate array type to be
198                                          saved
199 D820 F5                 PUSH  PSW        Save type array
200 D821 C226D8             JNZ   :D826      Jump if INT/FPT array
201 D824 E7                 RST   4          Move stringarray into one
202 D825 75                 DATA  :75        string in free RAM
203 D826 F1         LD99    POP   PSW        Get type array
204 D827 E5                 PUSH  H
205 D828 D5                 PUSH  D
206 D829 F5                 PUSH  PSW
207 D82A CD91E7             CALL  :E791      (0) Evaluate program name
208 D82D 3E32               MVI   A,:32      File type byte '2'
209 D82F CDC502             CALL  :02C5      WOPEN
210 D832 F1                 POP   PSW        Get type array
211 D833 213E01             LXI   H,:013E    Startaddr EBUF
212 D836 77                 MOV   M,A        Type into EBUF
213 D837 C368D6             JMP   :D668      Write 2 blocks + trailer
214                 *
215 D83A 00                 NOP
216                 *
217                 ****************************************
218                 * EVALUATE ARRAY TYPE TO BE SAVED/LOADED *
219                 ****************************************
220                 *
221                 * Exit: A:   Array type.
222                 *       DE:  Length all array elements.
223                 *       HL:  Beginaddr. 1st array element.
224                 *       Z=1: String array.
225                 *       Z=0: INT/FPT array.
226                 *
227 D83B CD5AE9     RLSAS   CALL  :E95A      (0) Get array addr in HL
228 D83E E630               ANI   :30        Allow any type array
229 D840 F5                 PUSH  PSW        Save type
230 D841 5E                 MOV   E,M        )
231 D842 23                 INX   H          ) Get array pntr in DE
232 D843 56                 MOV   D,M        )
233 D844 7B                 MOV   A,E
234 D845 B2                 ORA   D
235 D846 CA90E9             JZ    :E990      (0) Undef. array if no addr
236 D849 EB                 XCHG             Pointer to array in HL
237 D84A 2B                 DCX   H
238 D84B 2B                 DCX   H
239 D84C 56                 MOV   D,M        Get 1st length byte
240 D84D 23                 INX   H
241 D84E 7E                 MOV   A,M        Get 2nd length byte
242 D84F 23                 INX   H
243 D850 96                 SUB   M          Minus nr of dim. bytes
244 D851 5F                 MOV   E,A
245 D852 7A                 MOV   A,D
246 D853 DE00               SBI   :00        Update hibyte if borrow
247 D855 57                 MOV   D,A
248 D856 1B                 DCX   D          DE is length all array elem.
249 D857 CD39DE             CALL  :DE39      HL is beginaddr 1st element
```



```
250 D85A F1                 POP   PSW        Get type in A
251 D85B FE20               CPI   :20        Set Z-flag on type
252 D85D C9                 RET
253                 *
254                 **********************
255                 * RUN basiccmd LOADA *
256                 **********************
257                 *
258 D85E CD3BD8     RLODA   CALL  :D83B      Evaluate array type to be
259                                          loaded
260 D861 E5                 PUSH  H          Save startaddr array elem.
261 D862 F5                 PUSH  PSW        Save type
262 D863 CD78D6             CALL  :D678      Evaluate requested program
263                                          name; prep. select ROM bank
264 D866 CD08D8             CALL  :D808      Select ROM bank 1
265 D869 F1                 POP   PSW        Get type
266 D86A C30FEE             JMP   :EE0F      (1) Read block from tape and
267                                          store it in array
268                 *
269                 ************************************
270                 * INITIALISE 'EDIT': EMPTY HEAP,   *
271                 * CLEAR SYMBOL TABLE, MOVE PROGRAM *
272                 ************************************
273                 *
274                 * Part of 'Init. EDIT' (0E265).
275                 *
276                 * Entry: CY=1: Not sufficient memory available.
277                 *
278 D86D DA10DA     MPT33   JC    :DA10      Evt. run error 'OUT OF
279                                          MEMORY'
280 D870 C323CB             JMP   :CB23      Empty heap, move
281                                          program, clear symtab
282                 *
283                 ******************************************
284                 * LIST CURRENT LINE IF LINENR IS CORRECT *
285                 ******************************************
286                 *
287                 * Part of 'run LIST <range>' (0E1B6).
288                 *
289                 * Entry: CY=0 if linenr. <= 0 or > FFFF.
290                 *
291 D873 D215DA     MPT3A   JNC   :DA15      Evt. run error 'NUMBER
292                                          OUT OF RANGE'
293 D876 C3ABEC             JMP   :ECAB      (0) List current line
294                 *
295                 ***************************
296                 * INPUT FROM EDIT BUFFER *
297                 ***************************
298                 *
299                 * Part of 'restart interpreter' (C823).
300                 *
301 D879 E5         MPT05   PUSH  H
302 D87A CD91E2             CALL  :E291      (0) Input from editbuffer
303 D87D E1                 POP   H
304 D87E C9                 RET
305                 *
306                 ********************************
307                 * part of RUN 'CLEAR' (0E6B5) *
308                 ********************************
309                 *
310                 * Checks for heap too big.
311                 *
```

```
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
```

```
436 D8EA 7A                 MOV   A,D       Busaddress in A
437 D8EB 3201FE             STA   :FE01     Store busaddress in PB
438 D8EE 34                 INR   M         Set bus expand signal
439 D8EF 36FB               MVI   M,:FB     Set read strobe true
440                                         (Now data exchange)
441 D8F1 3A00FE             LDA   :FE00     Data to A
442 D8F4 5F                 MOV   E,A       Data in E
443 D8F5 36FF               MVI   M,:FF     Reset strobe
444 D8F7 35                 DCR   M
445 D8F8 E1                 POP   H
446 D8F9 F1                 POP   PSW
447 D8FA C9                 RET
448                 *
449                 *
450                 *
451 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   D8C8
SCN30  D89E   SNDINI D8A6   SNI10  D8C0   SPT04  D886
WPT    D7CB
```



```
002                         ORG   :D8FB
003                 *
004                 *
005                 *
006                 *  ===================
007                 *** INTERRUPT HANDLER ***
008                 *  ===================
009                 *
010                 *
011                 ********************************
012                 * INITIALISE INTERRUPT SYSTEM *
013                 ********************************
014                 *
015                 * Initialises the interrupt system of the machine.
016                 *
017                 * Entry: No conditions.
018                 * Exit:  All registers corrupted.
019                 *
020 D8FB 21F8FF     INTINI  LXI   H,:FFF8   Address int.mask register
021 D8FE 3E04               MVI   A,:04
022 D900 77                 MOV   M,A       Only stack interrupt
023 D901 325F00             STA   :005F     Remember int. mask
024 D904 2EF4               MVI   L,:F4
025 D906 3E0C               MVI   A,:0C
026 D908 77                 MOV   M,A       Select ext.int. and INTA
027 D909 3C                 INR   A
028 D90A 77                 MOV   M,A       Reset
029 D90B 3E0C               MVI   A,:0C
030 D90D 32C001             STA   :01C0     Init. cursor timer
031 D910 3E02               MVI   A,:02
032 D912 32C101             STA   :01C1     Init. keyboard scan counter
033 D915 CDA3D9             CALL  :D9A3     Reload sound timer
034 D918 CD9DD9             CALL  :D99D     Reload keyboard timer
035 D91B CD49D9             CALL  :D949     Set up int. entry points
036
037                 * Set-up interrupt vectors (also entry from utility)
038
039 D91E 21A9D9     INTSU   LXI   H,:D9A9
040 D921 227000             SHLD  :0070     Clock int. vector (7)
041 D924 2178D5             LXI   H,:D578
042 D927 226E00             SHLD  :006E     Keyboard int. vector (6)
043 D92A 21FDC6             LXI   H,:C6FD
044 D92D 226C00             SHLD  :006C     Screen restart vector (5)
045 D930 21C0C6             LXI   H,:C6C0
046 D933 226A00             SHLD  :006A     Math. restart vector (4)
047 D936 2155D7             LXI   H,:D755
048 D939 226800             SHLD  :0068     Sound int. vector (3)
049 D93C 21E2D9             LXI   H,:D9E2
050 D93F 226600             SHLD  :0066     Stack int. vector (2)
051 D942 210EC7             LXI   H,:C70E
052 D945 226400             SHLD  :0064     Utility/encode vector (1)
053 D948 C9                 RET
054                 *
055                 * SET UP INTERRUPT VECTOR AREA:
056                 *
057                 * Sets up the vector area 0000-003F.
058                 *
059                 * Entry: No conditions.
060                 * Exit:  All registers corrupted.
061                 *
062 D949 010000     VECSU   LXI   B,:0000   Start at zero
063 D94C 116BD9     VCS10   LXI   D,:D96B   Startaddr. int. routine
```

```
064 D94F 2173D9             LXI   H,:D973   Endaddress
065 D952 CD4FDE             CALL  :DE4F     Transfer int. routine
066 D955 79                 MOV   A,C
067 D956 FE40               CPI   :40       8 routines done?
068 D958 C24CD9             JNZ   :D94C     Next one if not ready
069 D95B 213B00             LXI   H,:003B   Addr. highest int.vector
070 D95E 01F8FF             LXI   B,:FFF8
071 D961 1670               MVI   D,:70
072 D963 72          VCS20  MOV   M,D       Load vector addr.
073 D964 15                 DCR   D
074 D965 15                 DCR   D
075 D966 09                 DAD   B         Calculate next addr
076 D967 DA63D9             JC    :D963     Next one if not ready
077 D96A C9                 RET
078                  *
079                  * INTERRUPT VECTOR ROUTINE:
080                  *
081                  * During initialisation loaded into RAM on the
082                  * Restart routine locations of the CPU.
083                  * The address after LHLD is later changed into
084                  * the appropriate vector address by #D8FB.
085                  *
086 D96B 00           ITMPL  NOP
087 D96C E5                 PUSH  H
088 D96D 2A7000             LHLD  :0070
089 D970 E9                 PCHL
090 D971 00                 NOP
091 D972 00                 NOP
092                  *
093                  ********************************
094                  * DISABLE KEYBOARD INTERRUPTS *
095                  ********************************
096                  *
097                  * Disables keyboard interrupts only.
098                  *
099                  * Entry can be different. From INTCH common part
100                  * to disable/enable sound, clock and keyboard
101                  * interrupts.
102                  *
103                  * Entry: None.
104                  * Exit:  All registers preserved.
105                  *
106 D973 C5          KBDI   PUSH  B
107 D974 0100BF             LXI   B,:BF00   Disable keyboard interrupts
108                  *
109 D977 F5          INTCH  PUSH  PSW
110 D978 F3                 DI              Disable interrupts
111 D979 3A5F00             LDA   :005F     Get current int.mask
112 D97C A0                 ANA   B         ) Calculate new one
113 D97D B1                 ORA   C         )
114 D97E 325F00             STA   :005F     Remember new mask
115 D981 32F8FF             STA   :FFF8     and set mask
116 D984 FB                 EI              Enable interrupts again
117 D985 F1                 POP   PSW
118 D986 C1                 POP   B
119 D987 C9                 RET
120                  *
121                  ********************************
122                  * ENABLE KEYBOARD INTERRUPTS *
123                  ********************************
124                  *
125                  * Enables keyboard interrupts only.
```



```
126                  *
127 D988 C5          KBEI   PUSH  B
128 D989 0140FF             LXI   B,:FF40   Enable keyboard interrupts
129 D98C C377D9             JMP   :D977     Update int. mask
130                  *
131                  ********************************
132                  * DISABLE SOUND INTERRUPTS *
133                  ********************************
134                  *
135                  * Disables sound interrupts only.
136                  *
137 D98F C5          SNDDI  PUSH  B
138 D990 0100F7             LXI   B,:F700   Disable sound interrupts
139 D993 C377D9             JMP   :D977     Update int. mask
140                  *
141                  ********************************
142                  * ENABLE SOUND INTERRUPTS *
143                  ********************************
144                  *
145                  * Enables sound interrupts only.
146                  *
147 D996 C5          SNDEI  PUSH  B
148 D997 0108FF             LXI   B,:FF08   Enable sound interrupts
149 D99A C377D9             JMP   :D977     Update int. mask
150                  *
151                  ************************
152                  * LOAD KEYBOARD TIMER *
153                  ************************
154                  *
155                  * Starts a keyboard interrupt.
156                  *
157 D99D 3EFF        KBIS   MVI   A,:FF      Init. 16.32 msec
158 D99F 32FCFF             STA   :FFFC     Load timer 4 (keyboard)
159 D9A2 C9                 RET
160                  *
161                  ********************
162                  * LOAD SOUND TIMER *
163                  ********************
164                  *
165                  * Starts a sound interrupt.
166                  *
167 D9A3 3E50        SNDIS  MVI   A,:50      Init. 5.12 msec
168 D9A5 32FBFF             STA   :FFFB     Load timer 3 (sound)
169 D9A8 C9                 RET
170                  *
171                  ****************************
172                  * CLOCK INTERRUPT (RST 7) *
173                  ****************************
174                  *
175                  * Triggered every 20 msec by the TV page
176                  * blanking signal.
177                  * Decrements timer 01BE/BF until 0. Checks also
178                  * the contents of cursor clock timer 01C0. It is
179                  * decremented; if it becomes 0, the timer is re-
180                  * loaded and the cursor is flashed.
181                  *
182                  * Exit: All registers preserved.
183                  *
184 D9A9 F5          CLKINT PUSH  PSW
185 D9AA C5                 PUSH  B
186 D9AB D5                 PUSH  D
187 D9AC 3A5F00             LDA   :005F     Get current int. mask
```

```
D9AF F5              PUSH  PSW       and remember it
D9B0 3E04            MVI   A,:04
D9B2 32F8FF          STA   :FFF8     Set stack interrupt only
D9B5 FB              EI
D9B6 2ABE01          LHLD  :01BE     Get timer contents
D9B9 7C              MOV   A,H
D9BA B5              ORA   L
D9BB CAC2D9          JZ    :D9C2     If timer = 0
D9BE 2B              DCX   H         Else:
D9BF 22BE01          SHLD  :01BE     decrement timer
D9C2 21C001  CKI10   LXI   H,:01C0   Addr. cursor clock
D9C5 35              DCR   M         Decr. clock
D9C6 C2CDD9          JNZ   :D9CD     Return if <>0
D9C9 360F            MVI   M,:0F     Load 20 ms flash time
D9CB EF              RST   5         Flash cursor
D9CC 12              DATA  :12
                     *
                     * GENERAL INTERRUPT RETURN:
                     *
                     * Restores interrupt mask and all registers.
                     *
                     * Entry: Interrupt mask and all registers on stack.
                     *
D9CD F1      INTRM   POP   PSW       Get old int. mask
D9CE F3              DI
D9CF 32F8FF          STA   :FFF8     Restore int. mask
D9D2 325F00          STA   :005F     and remember it.
D9D5 FB              EI
D9D6 D1              POP   D
D9D7 C1              POP   B
D9D8 F1              POP   PSW
D9D9 E1              POP   H
D9DA C9              RET
                     *
                     ***************************
                     * ENABLE CLOCK INTERRUPT *
                     ***************************
                     *
D9DB C5      CLKEI   PUSH  B
D9DC 0180FF          LXI   B,:FF80   Enable clock interrupt
D9DF C377D9          JMP   :D977     Update int.mask
                     *
                     ***************************
                     * STACK INTERRUPT (RST2) *
                     ***************************
                     *
                     * This routine handles an interrupt caused by the
                     * stack overflow hardware logic.
                     *
D9E2 3100F9  SPINT   LXI   SP,:F900  Reset stackpointer
D9E5 AF              XRA   A
D9E6 321701          STA   :0117     No running inputs
D9E9 322201          STA   :0122     No encoding of stored line
D9EC CD43D7          CALL  :D743     Input from keyboard, reload
                                     timers sound/keyb
D9EF FB              EI
D9F0 3E16            MVI   A,:16
D9F2 C3F5D9          JMP   :D9F5     Run error 'STACK OVERFLOW'
                     *
                     *
                     *
                     *
```



```
                     *  ===============
                     *** ERROR HANDLER ***
                     *  ===============
                     *
                     *
                     ******************
                     * ERROR HANDLING *
                     ******************
                     *
                     * Produces an error message and other information
                     * about errors.
                     *
                     * Entry: A:  Error message number.
                     *        BC: Latest position in EBUF or textbuffer.
                     * Exit:  If during input: Restart input statement.
                     *        If during encoding: Back to ELINA (C93C)
                     *                            for handling.
                     *        Else: Restart Basic (direct mode).
                     *
D9F5 47      ERROR   MOV   B,A       Save error pointer
D9F6 AF              XRA   A
D9F7 323101          STA   :0131     Set output screen/RS232
D9FA CDE4CE          CALL  :CEE4     Select ROM bank 0
D9FD 0000            DBL   :0000
D9FF 00              NOP
DA00 00              NOP
DA01 3A1801          LDA   :0118     Get RUNF
DA04 B7              ORA   A
DA05 C23DDA          JNZ   :DA3D     If run-time error
DA08 C364DA          JMP   :DA64     If compile-time error
                     *
                     * ENTRYPOINTS TO ERROR ROUTINES:
                     *
DA0B 3E17    ERRSN   MVI   A,:17
DA0D C3F5D9          JMP   :D9F5     Run 'SYNTAX ERROR'
                     *
DA10 3E13    ERROM   MVI   A,:13
DA12 C3F5D9          JMP   :D9F5     Run 'OUT OF MEMORY'
                     *
DA15 3E09    ERRRA   MVI   A,:09
DA17 C3F5D9          JMP   :D9F5     Run 'NUMBER OUT OF RANGE'
                     *
DA1A 3E14    ERRTM   MVI   A,:14
DA1C C3F5D9          JMP   :D9F5     Run 'TYPE MISMATCH'
                     *
DA1F 3E03    ERROV   MVI   A,:03
DA21 C3F5D9          JMP   :D9F5     Run 'OVERFLOW'
                     *
DA24 3E06    ERRDO   MVI   A,:06
DA26 C3F5D9          JMP   :D9F5     Run 'DIVISION BY ZERO'
                     *
DA29 3E05    ERRBS   MVI   A,:05
DA2B C3F5D9          JMP   :D9F5     Run 'SUBSCRIPT ERROR'
                     *
DA2E 3E00    ERRNF   MVI   A,:00
DA30 C3F5D9          JMP   :D9F5     Run 'NEXT WITHOUT FOR'
                     *
DA33 3E12    ERREL   MVI   A,:12
DA35 C3F5D9          JMP   :D9F5     Run 'ERROR LINE RUN'
                     *
DA38 3E08    ERRLS   MVI   A,:08
DA3A C3F5D9          JMP   :D9F5     Run 'STRING TOO LONG'
```

```
                        *
                        ******************
                        * RUN-TIME ERROR *
                        ******************
                        *
DA3D CD50DA     ERRRU   CALL  :DA50     Print error message
DA40 3A1701             LDA   :0117     Get RDIPL
DA43 B7                 ORA   A
DA44 C24DDA             JNZ   :DA4D     Jump if running inputs
DA47 CD75DA             CALL  :DA75     else: Print 'IN LINE <nr>'
DA4A C314C8             JMP   :C814     Re-enter BASIC

                        * If error in input:

DA4D C350E3     ERT10   JMP   :E350     (0) Back to restart input
                        *
                        ************************
                        * PRINT ERROR MESSAGE *
                        ************************
                        *
                        * Entry: B: Error message number.
                        * Exit:  BC preserved, AFDEHL corrupted.
                        *
DA50 CD55DD     ERRMS   CALL  :DD55     Cursor to begin (next) line
DA53 78                 MOV   A,B       Get error number
DA54 2194DA             LXI   H,:DA94   Base of list error messages
DA57 87                 ADD   A         Multiply error nr *2
DA58 5F                 MOV   E,A       and save offset
DA59 1600               MVI   D,:00
DA5B 19                 DAD   D         Calculate table addr.
DA5C 5E                 MOV   E,M       ) Get addr of message
DA5D 23                 INX   H         ) in DE
DA5E 56                 MOV   D,M       )
DA5F EB                 XCHG            and in HL
DA60 CDD4DA             CALL  :DAD4     Print error message
DA63 C9                 RET
                        *
                        ***********************
                        * COMPILE-TIME ERROR *
                        ***********************
                        *
                        * Handles errors in direct mode. Only an error
                        * message is printed. Control is passed back to
                        * ELINA (C93C) except if it is an error during
                        * encoding of a stored line.
                        *
                        * Entry: A: Error number.
                        *        C: Points to last read character on
                        *           input line.
                        * Exit:  To Basic interpreter.
                        *
DA64 3A2201     ERRCO   LDA   :0122     Get ERSFL
DA67 FE01               CPI   :01       Error during encoding?
DA69 CA57C9             JZ    :C957     Then handle error

                        * Error in direct command:

DA6C CD50DA             CALL  :DA50     Print error message
DA6F CD5EDD             CALL  :DD5E     Print car.ret
DA72 C323C8             JMP   :C823     Restart interpreter
                        *
                        *
```



```
                        ********************************************
                        * PRINT LINE NUMBER IN WHICH ERROR OCCURED *
                        ********************************************
                        *
                        * Prints car.ret if current line is a direct command
                        * Else, prints 'IN LINE <linenr>' and car.ret.
                        *
                        * Entry: None.
                        * Exit:  ABCDEHL preserved. F corrupted.
                        *        Z=1: direct command.
                        *
DA75 E5         MSGIL   PUSH  H
DA76 F5                 PUSH  PSW
DA77 2A0001             LHLD  :0100     Get start current line
DA7A 7C                 MOV   A,H
DA7B B5                 ORA   L         Check if 0
DA7C F5                 PUSH  PSW
DA7D CA8CDA             JZ    :DA8C     If direct, print car.ret
DA80 CDFFDA             CALL  :DAFF     Else, print 'IN LINE'
DA83 7FDB               DBL   :DB7F
DA85 7E                 MOV   A,M       ) Get line nr in HL
DA86 23                 INX   H         )
DA87 6E                 MOV   L,M       )
DA88 67                 MOV   H,A       )
DA89 CDB4EF             CALL  :EFB4     (0) Print line number
DA8C CD5EDD     MIL10   CALL  :DD5E     Print car.ret
DA8F F1                 POP   PSW
DA90 E1                 POP   H
DA91 7C                 MOV   A,H
DA92 E1                 POP   H
DA93 C9                 RET
                        *
                        ************************************
                        * ERROR MESSAGES INDIRECTION TABLE *
                        ************************************
                        *
                        * The address points to the location where the
                        * string with the error messages can be found.
                        * Between brackets the error number.
                        *
                        * Run-time errors:
                        ERITB
DA94 1CDC               E?NF    DBL   :DC1C     (00) NEXT WITHOUT FOR
DA96 2CDC               E?RG    DBL   :DC2C     (01) RETURN WITHOUT GOSUB
DA98 33DC               E?OD    DBL   :DC33     (02) OUT OF DATA
DA9A 3EDC               E?OV    DBL   :DC3E     (03) OVERFLOW
DA9C 50DC               E?US    DBL   :DC50     (04) UNDEFINED LINE NUMBER
DA9E 5CDC               E?BS    DBL   :DC5C     (05) SUBSCRIPT ERROR
DAA0 68DC               E?DO    DBL   :DC68     (06) DIVISION BY ZERO
DAA2 95DC               E?OS    DBL   :DC95     (07) OUT OF STRING SPACE
DAA4 9DDC               E?LS    DBL   :DC9D     (08) STRING TOO LONG
DAA6 D8DC               E?RA    DBL   :DCD8     (09) NUMBER OUT OF RANGE
DAA8 B2DC               E?IN    DBL   :DCB2     (0A) INVALID NUMBER
DAAA FADC               E?LO0   DBL   :DCFA     (0B) LOADING ERROR 0
DAAC FEDC               E?LO1   DBL   :DCFE     (0C) LOADING ERROR 1
DAAE 02DD               E?LO2   DBL   :DD02     (0D) LOADING ERROR 2
DAB0 06DD               E?LO3   DBL   :DD06     (0E) LOADING ERROR 3
DAB2 F1DC               E?UA    DBL   :DCF1     (0F) UNDEFINED ARRAY
DAB4 C2DC               E?NC    DBL   :DCC2     (10) COLOUR NOT AVAILABLE
DAB6 B7DC               E?OF    DBL   :DCB7     (11) OFF SCREEN
DAB8 12DD               E?EL    DBL   :DD12     (12) ERROR LINE RUN
```

```
436                     * Compile/Run-time errors:
437
438 DABA 47DC         E?OM    DBL    :DC47      (13) OUT OF MEMORY
439 DABC 8ADC         E?TM    DBL    :DC8A      (14) TYPE MISMATCH
440 DABE D6DC         E?LN    DBL    :DCD6      (15) LINE NUMBER OUT OF
441                                               .    RANGE
442 DAC0 38DC         E?SO    DBL    :DC38      (16) STACK OVERFLOW
443
444                     * Compile-time errors:
445
446 DAC2 23DC         E?SN    DBL    :DC23      (17) SYNTAX ERROR
447 DAC4 79DC         E?ID    DBL    :DC79      (18) COMMAND INVALID
448 DAC6 A9DC         E?CN    DBL    :DCA9      (19) CAN'T CONT
449 DAC8 E3DC         E?TC    DBL    :DCE3      (1A) LINE TOO COMPLEX
450 DACA 47DC         E?ST    DBL    :DC47      (1B) OUT OF MEMORY
451                     *
452                     ********************************
453                     * WAIT; part of RTALK (0EC6D) *
454                     ********************************
455                     *
456                     * Entry: HL: Wait time.
457                     *
458 DACC 2B           RTK20   DCX    H          Wait time -1
459 DACD 7C                   MOV    A,H
460 DACE B5                   ORA    L
461 DACF C2CCDA               JNZ    :DACC      If not ready
462 DAD2 EB                   XCHG              Parameter pntr in HL
463 DAD3 C9                   RET
464                     *
465                     *
466                     *
467 DAD4                      END

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

CKI10   D9C2    CLKEI   D9DB    CLKINT D9A9    E?BS    DA9E
E?CN    DAC6    E?DO    DAA0    E?EL   DAB8    E?ID    DAC4
E?IN    DAA8    E?LN    DABE    E?LO0  DAAA    E?LO1   DAAC
E?LO2   DAAE    E?LO3   DAB0    E?LS   DAA4    E?NC    DAB4
E?NF    DA94    E?OD    DA98    E?OF   DAB6    E?OM    DABA
E?OS    DAA2    E?OV    DA9A    E?RA   DAA6    E?RG    DA96
E?SN    DAC2    E?SO    DAC0    E?ST   DACA    E?TC    DAC8
E?TM    DABC    E?UA    DAB2    E?US   DA9C    ERITB   DA94
ERRBS   DA29    ERRCD   DA64    ERRDO  DA24    ERREL   DA33
ERRLS   DA38    ERRMS   DA50    ERRNF  DA2E    ERROM   DA10
ERROR   D9F5    ERROV   DA1F    ERRRA  DA15    ERRRU   DA3D
ERRSN   DA0B    ERRTM   DA1A    ERT10  DA4D    INTCH   D977
INTINI  D8FB    INTRM   D9CD    INTSU  D91E    ITMPL   D96B
KBDI    D973    KBEI    D988    KBIS   D99D    MIL10   DA8C
MSGIL   DA75    RTK20   DACC    SNDDI  D98F    SNDEI   D996
SNDIS   D9A3    SPINT   D9E2    VCS10  D94C    VCS20   D963
VECSU   D949
```



```
002                             ORG    :DAD4
003                     *
004                     *
005                     *
006                     *  ================
007                     *** PRINT ROUTINES ***
008                     *  ================
009                     *
010                     *
011                     ******************
012                     * PRINT MESSAGE *
013                     ******************
014                     *
015                     * Prints a message, which may include internal
016                     * references to other submessages.
017                     *
018                     * Entry: Pointer to message in HL.
019                     * Exit:  Pointer after string in HL. Other
020                     *        registers preserved.
021                     *
022                     * Message format: A series of bytes:
023                     *         00     End of string.
024                     *         01-7F Printed character.
025                     *         >= 80 bit 14 set:
026                     *                bits 0-13 are offset in program of
027                     *                a message (char. terminated by a 0).
028                     *                bit 14 unset:
029                     *                bits 0-13 are offset in program of
030                     *                a string (1 byte length + char.).
031                     *
032 DAD4 F5           PMSG    PUSH   PSW
033 DAD5 7E           PMS10   MOV    A,M         Get character
034 DAD6 23                   INX    H           Points to next char.
035 DAD7 B7           PMS15   ORA    A           Check char.
036 DAD8 CAE4DA               JZ     :DAE4       If end message
037 DADB FAE6DA               JM     :DAE6       If submessage reference
038 DADE CD60DD               CALL   :DD60       Print character in A
039 DAE1 C3D5DA               JMP    :DAD5       Next
040                     *
041 DAE4 F1           PMS20   POP    PSW         End message
042 DAE5 C9                   RET
043
044                     * Submessage reference:
045
046 DAE6 FEC0         PMS30   CPI    :C0
047 DAE8 E5                   PUSH   H
048 DAE9 6E                   MOV    L,M         Lobyte in L
049 DAEA DAF6DA               JC     :DAF6       If reference to string
050 DAED 67                   MOV    H,A         BASE is at location C000
051 DAEE CDD4DA               CALL   :DAD4       Print submessage
052 DAF1 E1           PMS35   POP    H
053 DAF2 23                   INX    H
054 DAF3 C3D5DA               JMP    :DAD5       Next character
055
056                     * String reference:
057
058 DAF6 C640         PMS40   ADI    :40         (BASE SHR 8) - #80
059 DAF8 67                   MOV    H,A         Hibyte in H
060 DAF9 CD32DB               CALL   :DB32       Print substring pntd by HL
061 DAFC C3F1DA               JMP    :DAF1       Next character
062                     *
063                     *
```

```
                        ********************************************
                        * PRINT MESSAGE POINTED TO BY NEXT 2 BYTES *
                        ********************************************
                        *
                        * Entry: Top of stack points to the address where
                        *        the address of the message can be found.
                        * Exit:  AFBCDEHL preserved.
                        *        Returnaddress on stack.
                        *
DAFF E3                 PMSGR   XTHL            Get pntr from stack
DB00 D5                         PUSH  D
DB01 5E                         MOV   E,M       Get lobyte address
DB02 23                         INX   H
DB03 56                         MOV   D,M       Get hibyte address
DB04 23                         INX   H
DB05 EB                         XCHG            Addr message in HL
DB06 CDD4DA                     CALL  :DAD4     Print message
DB09 EB                         XCHG            HL pnts after message pntr
DB0A D1                         POP   D
DB0B E3                         XTHL            Restore stack
DB0C C9                         RET
                        *
                        ************************
                        * CURSOR TO NEXT FIELD *
                        ************************
                        *
                        * Part of 'run PRINT' (0E2B3).
                        * Moves the cursor to a new number output field.
                        * The field size is 12 character positions; field
                        * positions: 0,12,24,36,48.
                        *
DB0D EF                 PSKP    RST   5         Ask cursor position
DB0E 0C                         DATA  :0C       and size character screen
DB0F 7D                         MOV   A,L       X-coord in L
DB10 FE30                       CPI   :30       Already past last field?
DB12 D227DB                     JNC   :DB27     Then print car.ret, abort
DB15 D60C               PSK10   SUI   :0C       ) Minus 12 untill underflow
DB17 D215DB                     JNC   :DB15     )
DB1A 2F                 PSK15   CMA             Restore pos. value
DB1B 3C                         INR   A

                        * Entry from 'SPC' function:

DB1C 57                 PSK20   MOV   D,A       Store nr of spaces required
DB1D 3E20               PSK30   MVI   A,:20
DB1F CD60DD                     CALL  :DD60     Print space
DB22 15                         DCR   D         Ready?
DB23 C21DDB                     JNZ   :DB1D     Next space if not
DB26 C9                         RET
                        *
DB27 C35EDD             PSK40   JMP   :DD5E     Print car.ret
                        *
                        ********************
                        * CURSOR TO TAB(8) *
                        ********************
                        *
                        * Part of 'List current line' (0ECB3).
                        * Moves cursor to column 8 after linenumber.
                        *
                        * Exit: BC preserved. AFDEHL corrupted.
                        *
                        *
```



```
                        SCTAB
DB2A EF                 PTAB    RST   5         Ask cursor position and
DB2B 0C                         DATA  :0C       size character screen
DB2C 7D                         MOV   A,L       X-coord after linenr in A
DB2D D606                       SUI   :06       Tab must be 8
DB2F C31ADB                     JMP   :DB1A     Print additional spaces
                        *
                        ****************
                        * PRINT STRING *
                        ****************
                        *
                        * Prints a string of characters pointed to by HL.
                        *
                        * Entry: HL points to string.
                        * Exit:  HL points after string.
                        *        Other registers preserved.
                        *
                        * String format:
                        *        1 byte length (0 = no data).
                        *        N bytes data.
                        *
                        SCSTR
DB32 F5                 PSTR    PUSH  PSW
DB33 C5                         PUSH  B
DB34 46                         MOV   B,M       String length in B
DB35 23                 PST10   INX   H
DB36 78                 PST20   MOV   A,B       Get length
DB37 D601                       SUI   :01       Minus 1
DB39 47                         MOV   B,A       Nr. char still to be printed
DB3A 7E                         MOV   A,M       Get char
DB3B D495D6                     CNC   :D695     Print character if not ready
DB3E D235DB                     JNC   :DB35     Next one if not ready
DB41 C1                         POP   B
DB42 F1                         POP   PSW
DB43 C9                         RET
                        *
                        ************************
                        * PRINT STRING MESSAGE *
                        ************************
                        *
                        * Prints a string of characters, pointed to by
                        * HL, length in A.
                        *
                        * Entry: HL: Points to string.
                        *        A:  Number of characters.
                        * Exit:  HL: Points after string.
                        *        AFBCDE preserved.
                        *
                        SCSTM
DB44 F5                 PSTRM   PUSH  PSW
DB45 C5                         PUSH  B
DB46 47                         MOV   B,A       String length in B
DB47 C336DB                     JMP   :DB36     Print string
                        *
                        **********************
                        * PRINT A HEX NUMBER *
                        **********************
                        *
                        * Converts MACC to hex in DECBUF and print it.
                        *
                        * Exit: HL points after string in DECBUF.
                        *       BCDE preserved. AF corrupted.
```

```
                        *
DB4A CD2DC0     PHEX    CALL  :C02D     Convert MACC for hex output
DB4D 2A33C0     PGP     LHLD  :C033     Get addr DECBUF
DB50 C332DB             JMP   :DB32     Print string in DECBUF
                        *
                        ***************************
                        * PRINT A INTEGER NUMBER *
                        ***************************
                        *
                        * Prints an integer number from MACC.
                        *
DB53 CD5FDB     PINT    CALL  :DB5F     Convert MACC for output
DB56 C34DDB             JMP   :DB4D     Print contents DECBUF
                        *
                        ***********************************
                        * PRINT A FLOATING POINT NUMBER *
                        ***********************************
                        *
                        * Prints a FPT number from MACC.
                        *
DB59 CD9BCE     PFPT    CALL  :CE9B     Convert MACC for output
DB5C C34DDB             JMP   :DB4D     Print contents DECBUF
                        *
                        ****************************************
                        * CONVERT MACC FOR FIXED POINT OUTPUT *
                        ****************************************
                        *
                        * Places ASCII-equivalent in 00E4-F0, digits before
                        * decimal point in 00F1, length in 00E3.
                        *
                        * Exit: A: Number of digits.
                        *       BCDEHL preserved.
                        *
DB5F CD27C0     IBCP    CALL  :C027     Convert INT for output
DB62 C5                 PUSH  B
DB63 0600               MVI   B,:00     Can trim last dec. place
DB65 CD30C0     BPP     CALL  :C030     Tidy up into external form
DB68 C1                 POP   B
DB69 C9                 RET
                        *
                        *********************************
                        * (Not used, replaced by CE9B) *
                        *********************************
                        *
DB6A 0601       LD216   MVI   B,:01     ) Part of a previous version
DB6C C364DB             JMP   :DB64     )
                        *
                        *
DB6F                    END

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

BPP     DB65    IBCP    DB5F    LD216   DB6A    PFPT    DB59
PGP     DB4D    PHEX    DB4A    PINT    DB53    PMS10   DAD5
PMS15   DAD7    PMS20   DAE4    PMS30   DAE6    PMS35   DAF1
PMS40   DAF6    PMSG    DAD4    PMSGR   DAFF    PSK10   DB15
PSK15   DB1A    PSK20   DB1C    PSK30   DB1D    PSK40   DB27
PSKP    DB0D    PST10   DB35    PST20   DB36    PSTR    DB32
PSTRM   DB44    PTAB    DB2A    SCSTM   DB44    SCSTR   DB32
SCTAB   DB2A
```



```
                        ORG   :DB6F
                        *
                        *
                        *
                        *********************************
                        * STRINGS FOR MACHINE MESSAGES *
                        *********************************
                        *
                        * The machine messages exist partly from strings,
                        * partly from subreferences to other strings.
                        * The subreferences can be:
                        *     - An address where another string can be found.
                        *     - An offset with base at C000 to the other
                        *       string.
                        * The messages are ended with 00.
                        * 20 is space, 0D is carriage return.
                        *
                        RMS01
DB6F 53         MSG01   DATA  :53       S
DB70 4F                 DATA  :4F       O
DB71 4D                 DATA  :4D       M
DB72 45                 DATA  :45       E
DB73 20                 DATA  :20
DB74 8CA5               DATA  :8C,:A5   INPUT
DB76 20                 DATA  :20
DB77 49                 DATA  :49       I
DB78 47                 DATA  :47       G
DB79 4E                 DATA  :4E       N
DB7A 4F                 DATA  :4F       O
DB7B 52                 DATA  :52       R
DB7C 45                 DATA  :45       E
DB7D 44                 DATA  :44       D
DB7E 00                 DATA  :00
                        *
DB7F 20         MSG02   DATA  :20
DB80 49                 DATA  :49       I
DB81 4E                 DATA  :4E       N
DB82 20                 DATA  :20
DB83 4C         MLINE   DATA  :4C       L
DB84 49                 DATA  :49       I
DB85 4E                 DATA  :4E       N
DB86 45                 DATA  :45       E
DB87 20                 DATA  :20
DB88 00                 DATA  :00
                        *
DB89 DC0D       MSG03   DATA  :DC,:0D   OUT OF
DB8B 8E56               DATA  :8E,:56   SPACE
DB8D 20                 DATA  :20
DB8E 8CD2               DATA  :8C,:D2   FOR
DB90 D88D               DATA  :D8,:8D   MODE
DB92 00                 DATA  :00
                        *
DB93 20         MSG04   DATA  :20
DB94 52                 DATA  :52       R
DB95 45                 DATA  :45       E
DB96 DBF7               DATA  :DB,:F7   TYPE
DB98 DB83       MLINR   DATA  :DB,:83   LINE
DB9A 0D                 DATA  :0D
DB9B 00                 DATA  :00
                        *
DB9C 0D         MSG15   DATA  :0D
DB9D 53                 DATA  :53       S
```

```
064 DB9E 45                 DATA  :45       E
065 DB9F 54                 DATA  :54       T
066 DBA0 20                 DATA  :20
067 DBA1 52                 DATA  :52       R
068 DBA2 45                 DATA  :45       E
069 DBA3 43                 DATA  :43       C
070 DBA4 4F                 DATA  :4F       O
071 DBA5 52                 DATA  :52       R
072 DBA6 44                 DATA  :44       D
073 DBA7 2C                 DATA  :2C       ,
074 DBA8 DBB0       MSG05   DATA  :DB,:B0   START TAPE
075 DBAA 2C                 DATA  :2C       ,
076 DBAB DBF7               DATA  :DB,:F7   TYPE
077 DBAD 8E56               DATA  :8E,:56   SPACE
078 DBAF 00                 DATA  :00
079                 *
080 DBB0 53         MSG06   DATA  :53       S
081 DBB1 54                 DATA  :54       T
082 DBB2 41                 DATA  :41       A
083 DBB3 52                 DATA  :52       R
084 DBB4 54                 DATA  :54       T
085 DBB5 DBFD               DATA  :DB,:FD   TAPE
086 DBB7 00                 DATA  :00
087                 *
088 DBB8 0D         MSG07   DATA  :0D
089 DBB9 2A                 DATA  :2A       *
090 DBBA 2A                 DATA  :2A       *
091 DBBB 2A                 DATA  :2A       *
092 DBBC DBE0               DATA  :DB,:E0   BREAK
093 DBBE 0D                 DATA  :0D
094 DBBF 00                 DATA  :00
095                 *
096 DBC0 20         MSG17   DATA  :20
097 DBC1 4F                 DATA  :4F       O
098 DBC2 4B                 DATA  :4B       K
099 DBC3 0D                 DATA  :0D
100 DBC4 00                 DATA  :00
101                 *
102 DBC5 0D         MSG09   DATA  :0D
103 DBC6 DBE0               DATA  :DB,:E0   BREAK
104 DBC8 00                 DATA  :00
105                 *
106 DBC9 8BCE       MSG10   DATA  :8B,:CE   STOP
107 DBCB 50                 DATA  :50       P
108 DBCC 45                 DATA  :45       E
109 DBCD 44                 DATA  :44       D
110 DBCE 00                 DATA  :00
111                 *
112 DBCF 8BD6       MSG11   DATA  :8B,:D6   END
113 DBD1 20                 DATA  :20
114 DBD2 50                 DATA  :50       P
115 DBD3 52                 DATA  :52       R
116 DBD4 4F                 DATA  :4F       O
117 DBD5 47                 DATA  :47       G
118 DBD6 52                 DATA  :52       R
119 DBD7 41                 DATA  :41       A
120 DBD8 4D                 DATA  :4D       M
121 DBD9 0D                 DATA  :0D
122 DBDA 00                 DATA  :00
123                 *
124 DBDB 20         MSG14   DATA  :20
125 DBDC 42                 DATA  :42       B
```



```
126 DBDD 41                 DATA  :41       A
127 DBDE 44                 DATA  :44       D
128 DBDF 00                 DATA  :00
129                 *
130 DBE0 42         MBREAK  DATA  :42       B
131 DBE1 52                 DATA  :52       R
132 DBE2 45                 DATA  :45       E
133 DBE3 41                 DATA  :41       A
134 DBE4 4B                 DATA  :4B       K
135 DBE5 00                 DATA  :00
136                 *
137 DBE6 20         MWITHO  DATA  :20
138 DBE7 57                 DATA  :57       W
139 DBE8 49                 DATA  :49       I
140 DBE9 54                 DATA  :54       T
141 DBEA 48                 DATA  :48       H
142 DBEB 4F                 DATA  :4F       O
143 DBEC 55                 DATA  :55       U
144 DBED 54                 DATA  :54       T
145 DBEE 20                 DATA  :20
146 DBEF 00                 DATA  :00
147                 *
148 DBF0 53         MSTRIN  DATA  :53       S
149 DBF1 54                 DATA  :54       T
150 DBF2 52                 DATA  :52       R
151 DBF3 49         MING    DATA  :49       I
152 DBF4 4E                 DATA  :4E       N
153 DBF5 47                 DATA  :47       G
154 DBF6 00                 DATA  :00
155                 *
156 DBF7 54         MTYPE   DATA  :54       T
157 DBF8 59                 DATA  :59       Y
158 DBF9 50                 DATA  :50       P
159 DBFA 45                 DATA  :45       E
160 DBFB 20                 DATA  :20
161 DBFC 00                 DATA  :00
162                 *
163 DBFD 20         MTAPE   DATA  :20
164 DBFE 54                 DATA  :54       T
165 DBFF 41                 DATA  :41       A
166 DC00 50                 DATA  :50       P
167 DC01 45                 DATA  :45       E
168 DC02 00                 DATA  :00
169                 *
170 DC03 55         MUNDF   DATA  :55       U
171 DC04 4E                 DATA  :4E       N
172 DC05 44                 DATA  :44       D
173 DC06 45                 DATA  :45       E
174 DC07 46                 DATA  :46       F
175 DC08 49                 DATA  :49       I
176 DC09 4E                 DATA  :4E       N
177 DC0A 45                 DATA  :45       E
178 DC0B 44                 DATA  :44       D
179 DC0C 00                 DATA  :00
180                 *
181 DC0D 4F         MOUTOF  DATA  :4F       O
182 DC0E 55                 DATA  :55       U
183 DC0F 54                 DATA  :54       T
184 DC10 20                 DATA  :20
185 DC11 4F                 DATA  :4F       O
186 DC12 46                 DATA  :46       F
187 DC13 20                 DATA  :20
```

```
188 DC14 00             DATA  :00
189                     *
190 DC15 20             MERROR  DATA  :20
191 DC16 45                     DATA  :45       E
192 DC17 52                     DATA  :52       R
193 DC18 52                     DATA  :52       R
194 DC19 4F                     DATA  :4F       O
195 DC1A 52                     DATA  :52       R
196 DC1B 00                     DATA  :00
197                     *
198                     ***************************
199                     * STRINGS ERROR MESSAGES *
200                     ***************************
201                     *
202                     * Comments: See strings machine messages (DB6F).
203                     *
204 DC1C 8CD9           ERMNF   DATA  :8C,:D9   NEXT
205 DC1E DBE6                   DATA  :DB,:E6   WITHOUT
206 DC20 8CD2                   DATA  :8C,:D2   FOR
207 DC22 00                     DATA  :00
208                     *
209 DC23 53             ERMSN   DATA  :53       S
210 DC24 59                     DATA  :59       Y
211 DC25 4E                     DATA  :4E       N
212 DC26 54                     DATA  :54       T
213 DC27 41                     DATA  :41       A
214 DC28 58                     DATA  :58       X
215 DC29 DC15                   DATA  :DC,:15   ERROR
216 DC2B 00                     DATA  :00
217                     *
218 DC2C 8BE8           ERMRG   DATA  :8B,:E8   RETURN
219 DC2E DBE6                   DATA  :DB,:E6   WITHOUT
220 DC30 8C01                   DATA  :8C,:01   GOSUB
221 DC32 00                     DATA  :00
222                     *
223 DC33 DC0D           ERMOD   DATA  :DC,:0D   OUT OF
224 DC35 8CAE                   DATA  :8C,:AE   DATA
225 DC37 00                     DATA  :00
226                     *
227 DC38 53             ERMSO   DATA  :53       S
228 DC39 54                     DATA  :54       T
229 DC3A 41                     DATA  :41       A
230 DC3B 43                     DATA  :43       C
231 DC3C 4B                     DATA  :4B       K
232 DC3D 20                     DATA  :20
233 DC3E 4F             ERMOV   DATA  :4F       O
234 DC3F 56                     DATA  :56       V
235 DC40 45                     DATA  :45       E
236 DC41 52                     DATA  :52       R
237 DC42 46                     DATA  :46       F
238 DC43 4C                     DATA  :4C       L
239 DC44 4F                     DATA  :4F       O
240 DC45 57                     DATA  :57       W
241 DC46 00                     DATA  :00
242                     *
243 DC47 DC0D           ERMOM   DATA  :DC,:0D   OUT OF
244 DC49 4D                     DATA  :4D       M
245 DC4A 45                     DATA  :45       E
246 DC4B 4D                     DATA  :4D       M
247 DC4C 4F                     DATA  :4F       O
248 DC4D 52                     DATA  :52       R
249 DC4E 59                     DATA  :59       Y
```



```
250 DC4F 00             DATA  :00
251                     *
252 DC50 DC03           ERMUS   DATA  :DC,:03   UNDEFINED
253 DC52 20                     DATA  :20
254 DC53 DB53           MLINN   DATA  :DB,:53   LINE
255 DC55 4E             MNUMBE  DATA  :4E       N
256 DC56 55                     DATA  :55       U
257 DC57 4D                     DATA  :4D       M
258 DC58 42                     DATA  :42       B
259 DC59 45                     DATA  :45       E
260 DC5A 52                     DATA  :52       R
261 DC5B 00                     DATA  :00
262                     *
263 DC5C 53             ERMBS   DATA  :53       S
264 DC5D 55                     DATA  :55       U
265 DC5E 42                     DATA  :42       B
266 DC5F 53                     DATA  :53       S
267 DC60 43                     DATA  :43       C
268 DC61 52                     DATA  :52       R
269 DC62 49                     DATA  :49       I
270 DC63 50                     DATA  :50       P
271 DC64 54                     DATA  :54       T
272 DC65 DC15                   DATA  :DC,:15   ERROR
273 DC67 00                     DATA  :00
274                     *
275 DC68 44             ERMDO   DATA  :44       D
276 DC69 49                     DATA  :49       I
277 DC6A 56                     DATA  :56       V
278 DC6B 49                     DATA  :49       I
279 DC6C 53                     DATA  :53       S
280 DC6D 49                     DATA  :49       I
281 DC6E 4F                     DATA  :4F       O
282 DC6F 4E                     DATA  :4E       N
283 DC70 20                     DATA  :20
284 DC71 42                     DATA  :42       B
285 DC72 59                     DATA  :59       Y
286 DC73 20                     DATA  :20
287 DC74 5A                     DATA  :5A       Z
288 DC75 45                     DATA  :45       E
289 DC76 52                     DATA  :52       R
290 DC77 4F                     DATA  :4F       O
291 DC78 00                     DATA  :00
292                     *
293 DC79 43             ERMID   DATA  :43       C
294 DC7A 4F                     DATA  :4F       O
295 DC7B 4D                     DATA  :4D       M
296 DC7C 4D                     DATA  :4D       M
297 DC7D 41                     DATA  :41       A
298 DC7E 4E                     DATA  :4E       N
299 DC7F 44                     DATA  :44       D
300 DC80 20                     DATA  :20
301 DC81 49             MINVAL  DATA  :49       I
302 DC82 4E                     DATA  :4E       N
303 DC83 56                     DATA  :56       V
304 DC84 41                     DATA  :41       A
305 DC85 4C                     DATA  :4C       L
306 DC86 49                     DATA  :49       I
307 DC87 44                     DATA  :44       D
308 DC88 20                     DATA  :20
309 DC89 00                     DATA  :00
310                     *
311 DC8A DBF7           ERMTM   DATA  :DB,:F7   TYPE
```

```
312 DC8C 4D                     DATA  :4D       M
313 DC8D 49                     DATA  :49       I
314 DC8E 53                     DATA  :53       S
315 DC8F 4D                     DATA  :4D       M
316 DC90 41                     DATA  :41       A
317 DC91 54                     DATA  :54       T
318 DC92 43                     DATA  :43       C
319 DC93 48                     DATA  :48       H
320 DC94 00                     DATA  :00
321                     *
322 DC95 DC0D           ERMOS   DATA  :DC,:0D   OUT OF
323 DC97 DBF0                   DATA  :DB,:F0   STRING
324 DC99 20                     DATA  :20
325 DC9A 8E56                   DATA  :8E,:56   SPACE
326 DC9C 00                     DATA  :00
327                     *
328 DC9D DBF0           ERMLS   DATA  :DB,:F0   STRING
329 DC9F 20                     DATA  :20
330 DCA0 54                     DATA  :54       T
331 DCA1 4F                     DATA  :4F       O
332 DCA2 4F                     DATA  :4F       O
333 DCA3 20                     DATA  :20
334 DCA4 4C                     DATA  :4C       L
335 DCA5 4F                     DATA  :4F       O
336 DCA6 4E                     DATA  :4E       N
337 DCA7 47                     DATA  :47       G
338 DCA8 00                     DATA  :00
339                     *
340 DCA9 43             ERMCN   DATA  :43       C
341 DCAA 41                     DATA  :41       A
342 DCAB 4E                     DATA  :4E       N
343 DCAC 27                     DATA  :27       '
344 DCAD 54                     DATA  :54       T
345 DCAE 20                     DATA  :20
346 DCAF 8BC6                   DATA  :8B,:C6   CONT
347 DCB1 00                     DATA  :00
348                     *
349 DCB2 DC81           ERMIN   DATA  :DC,:81   INVALID
350 DCB4 DC55                   DATA  :DC,:55   NUMBER
351 DCB6 00                     DATA  :00
352                     *
353 DCB7 4F             ERMOF   DATA  :4F       O
354 DCB8 46                     DATA  :46       F
355 DCB9 46                     DATA  :46       F
356 DCBA 20                     DATA  :20
357 DCBB 53                     DATA  :53       S
358 DCBC 43                     DATA  :43       C
359 DCBD 52                     DATA  :52       R
360 DCBE 45                     DATA  :45       E
361 DCBF 45                     DATA  :45       E
362 DCC0 4E                     DATA  :4E       N
363 DCC1 00                     DATA  :00
364                     *
365 DCC2 43             ERMNC   DATA  :43       C
366 DCC3 4F                     DATA  :4F       O
367 DCC4 4C                     DATA  :4C       L
368 DCC5 4F                     DATA  :4F       O
369 DCC6 52                     DATA  :52       R
370 DCC7 20                     DATA  :20
371 DCC8 4E                     DATA  :4E       N
372 DCC9 4F                     DATA  :4F       O
373 DCCA 54                     DATA  :54       T
```



```
374 DCCB 20                     DATA  :20
375 DCCC 41                     DATA  :41       A
376 DCCD 56                     DATA  :56       V
377 DCCE 41                     DATA  :41       A
378 DCCF 49                     DATA  :49       I
379 DCD0 4C                     DATA  :4C       L
380 DCD1 41                     DATA  :41       A
381 DCD2 42                     DATA  :42       B
382 DCD3 4C                     DATA  :4C       L
383 DCD4 45                     DATA  :45       E
384 DCD5 00                     DATA  :00
385                     *
386 DCD6 DB83           ERMLN   DATA  :DB,:83   LINE
387 DCD8 DC55           ERMNA   DATA  :DC,:55   NUMBER
388 DCDA 20                     DATA  :20
389 DCDB DC0D           MSGOR   DATA  :DC,:0D   OUT OF
390 DCDD 52                     DATA  :52       R
391 DCDE 41                     DATA  :41       A
392 DCDF 4E                     DATA  :4E       N
393 DCE0 47                     DATA  :47       G
394 DCE1 45                     DATA  :45       E
395 DCE2 00                     DATA  :00
396                     *
397 DCE3 DB83           ERMTC   DATA  :DB,:83   LINE
398 DCE5 54                     DATA  :54       T
399 DCE6 4F                     DATA  :4F       O
400 DCE7 4F                     DATA  :4F       O
401 DCE8 20                     DATA  :20
402 DCE9 43                     DATA  :43       C
403 DCEA 4F                     DATA  :4F       O
404 DCEB 4D                     DATA  :4D       M
405 DCEC 50                     DATA  :50       P
406 DCED 4C                     DATA  :4C       L
407 DCEE 45                     DATA  :45       E
408 DCEF 58                     DATA  :58       X
409 DCF0 00                     DATA  :00
410                     *
411 DCF1 DC03           ERMUA   DATA  :DC,:03   UNDEFINED
412 DCF3 20                     DATA  :20
413 DCF4 41                     DATA  :41       A
414 DCF5 52                     DATA  :52       R
415 DCF6 52                     DATA  :52       R
416 DCF7 41                     DATA  :41       A
417 DCF8 59                     DATA  :59       Y
418 DCF9 00                     DATA  :00
419                     *
420 DCFA DD0A           ERML0   DATA  :DD,:0A   LOADING ERROR
421 DCFC 30                     DATA  :30       0
422 DCFD 00                     DATA  :00
423                     *
424 DCFE DD0A           ERML1   DATA  :DD,:0A   LOADING ERROR
425 DD00 31                     DATA  :31       1
426 DD01 00                     DATA  :00
427                     *
428 DD02 DD0A           ERML2   DATA  :DD,:0A   LOADING ERROR
429 DD04 32                     DATA  :32       2
430 DD05 00                     DATA  :00
431                     *
432 DD06 DD0A           ERML3   DATA  :DD,:0A   LOADING ERROR
433 DD08 33                     DATA  :33       3
434 DD09 00                     DATA  :00
435                     *
```

```
436                     ERMLO
437 DD0A 8D1B           MSGL    DATA  :8D,:1B   LOAD
438 DD0C DBF3                  DATA  :DB,:F3   ING
439 DD0E DC15                  DATA  :DC,:15   ERROR
440 DD10 20                    DATA  :20
441 DD11 00                    DATA  :00
442                     *
443 DD12 DC15           ERMEL   DATA  :DC,:15   ERROR
444 DD14 20                    DATA  :20
445 DD15 DB83                  DATA  :DB,:83   LINE
446 DD17 8BF2                  DATA  :8B,:F2   RUN
447 DD19 00                    DATA  :00
448                     *
449                     *
450                     *
451 DD1A                       END
```

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

ERMBS	DC5C	ERMCN	DCA9	ERMDO	DC68	ERMEL	DD12
ERMID	DC79	ERMIN	DCB2	ERMLO	DCFA	ERML1	DCFE
ERML2	DD02	ERML3	DD06	ERMLN	DCD6	ERMLO	DD0A
ERMLS	DC9D	ERMNA	DCD8	ERMNC	DCC2	ERMNF	DC1C
ERMOD	DC33	ERMOF	DCB7	ERMOM	DC47	ERMOS	DC95
ERMOV	DC3E	ERMRG	DC2C	ERMSN	DC23	ERMSO	DC38
ERMTC	DCE3	ERMTM	DC8A	ERMUA	DCF1	ERMUS	DC50
MBREAK	DBE0	MERROR	DC15	MING	DBF3	MINVAL	DC81
MLINE	DB83	MLINN	DC53	MLINR	DB98	MNUMBE	DC55
MOUTOF	DC0D	MSG01	DB6F	MSG02	DB7F	MSG03	DB89
MSG04	DB93	MSG05	DBA8	MSG06	DBB0	MSG07	DBB8
MSG09	DBC5	MSG10	DBC9	MSG11	DBCF	MSG14	DBDB
MSG15	DB9C	MSG17	DBC0	MSGL	DD0A	MSGOR	DCDB
MSTRIN	DBF0	MTAPE	DBFD	MTYPE	DBF7	MUNDF	DC03
MWITHO	DBE6	RMS01	DB6F				


```
002                             ORG   :DD1A
003                     *
004                     *
005                     *
006                     ********************
007                     * INPUT TEXT LINE *
008                     ********************
009                     *
010                     * Part of 'restart interpreter' (C853).
011                     * Scans keyboard and reads in a line on the current
012                     * screen line, up until car.ret. First prints
013                     * car.ret and a prompt ('*').
014                     * The routine can be aborted on car.ret or Break
015                     * only.
016                     *
017                     * Entry: A: Contains prompt.
018                     * Exit:  CY=1: Break pressed.
019                     *        CY=0: HL:  Address 1st character on line.
020                     *              C=1: Offset 1st significant character
021                     *              ABDEHL preserved.
022                     *
023 DD1A F5             INPLO   PUSH  PSW
024 DD1B CD55DD                CALL  :DD55     Cursor to column 0
025 DD1E F1                    POP   PSW       Get prompt
026 DD1F 37             INPLN   STC             CY=1
027 DD20 F5                    PUSH  PSW
028 DD21 E5                    PUSH  H         Save cursor coord 1st char
029 DD22 CD6ADD         IPL10   CALL  :DD6A     Print prompt
030 DD25 21B902                LXI   H,:02B9
031 DD28 3600                  MVI   M,:00     Enable complete keyb.scan
032 DD2A CDBED6         IPL20   CALL  :D6BE     Get keyb. input
033 DD2D DA49DD                JC    :DD49     Ignore line if Break presse
034 DD30 CA2ADD                JZ    :DD2A     Wait for input
035 DD33 FE20                  CPI   :20       Printable character?
036 DD35 D222DD                JNC   :DD22     Print it and get next one
037 DD38 FE08                  CPI   :08       Backspace
038 DD3A CA22DD                JZ    :DD22     Print it; get next char
039 DD3D FE0D                  CPI   :0D       Car.ret?
040 DD3F C22ADD                JNZ   :DD2A     Get next char if not
041
042                     * Exit on car.ret:
043
044 DD42 35                    DCR   M         Set KBRFL for BREAK only
045 DD43 0E01                  MVI   C,:01
046 DD45 E1             EXIT1   POP   H         Get cursor coord 1st char
047 DD46 F1                    POP   PSW       Get prompt
048 DD47 3F                    CMC             CY=0
049 DD48 C9                    RET
050
051                     * Exit on Break:
052
053 DD49 35             IPL30   DCR   M         Set KBRFL for BREAK only
054 DD4A 3E21                  MVI   A,:21
055 DD4C CD60DD                CALL  :DD60     Print '!'
056 DD4F CD5EDD                CALL  :DD5E     Print car.ret
057 DD52 E1                    POP   H
058 DD53 F1                    POP   PSW       Get prompt, CY=1
059 DD54 C9                    RET
060                     *
061                     **********************
062                     * CURSOR TO COLUMN 0 *
063                     **********************
```

```
                        *
                        * The X-coordinate of the cursor is checked.
                        * If not 0, a car.ret is printed.
                        *
                        * Entry: None.
                        * Exit:  AF corrupted. BCDEHL preserved.
                        *
DD55 D5                 COLO    PUSH  D
DD56 E5                         PUSH  H
DD57 EF                         RST   5         Get cursor pos (HL) and size
DD58 0C                         DATA  :0C       char.screen (DE)
DD59 7D                         MOV   A,L       X-coord cursor in A
DD5A E1                         POP   H
DD5B D1                         POP   D
DD5C B7                         ORA   A
DD5D C8                         RZ              Abort if cursor in Col.0
DD5E 3E0D               CRLF    MVI   A,:0D     Else: print CR
                        *
                        * GENERAL OUTPUT ROUTINE:
                        *
                        * Outputs a character in a direction depending
                        * on OTSW (#0131).
                        *
                        * Entry: A: Character to be transmitted.
                        * Exit:  AF corrupted.
                        *
                        SCCHR
DD60 F5                 OUTC    PUSH  PSW       Preserve char
DD61 3A3101                     LDA   :0131
DD64 FE02                       CPI   :02       Check output direction
DD66 D270DD                     JNC   :DD70     If to edit buf/DOUTC
                        
                        * To screen/RS232 - OTSW=0/1:

DD69 F1                         POP   PSW       Get char
DD6A EF                 COUTC   RST   5         Character to screen
DD6B 03                         DATA  :03
DD6C D442D6                     CNC   :D642     Output to RS232 if reqd
DD6F C9                         RET

                        * To DOUTC - OTSW=3:

DD70 00                 OTC10   NOP
DD71 C24CD7                     JNZ   :D74C     Character to DOUTC

                        * To editbuffer - OTSW=2:

DD74 F1                         POP   PSW       Get char
DD75 F5                 OTBIN   PUSH  PSW
DD76 E5                         PUSH  H
DD77 D5                         PUSH  D
DD78 2AA400                     LHLD  :00A4     Get edit input pointer
DD7B 77                         MOV   M,A       Byte in edit buffer
DD7C 23                         INX   H
DD7D 22A400                     SHLD  :00A4     Update edit input pointer
DD80 EB                         XCHG            in DE
DD81 2AA600                     LHLD  :00A6     Get end edit buffer
DD84 CD14DE                     CALL  :DE14     Calculate free buffer space
DD87 DA8EDD                     JC    :DD8E     If edit buffer full
DD8A D1                         POP   D
DD8B E1                         POP   H
DD8C F1                 OTC99   POP   PSW
```



```
DD8D C9                         RET

                        * If edit buffer full:

DD8E CDCADE             OTC20   CALL  :DECA     Heap back to right size
DD91 C310DA                     JMP   :DA10     Run error 'OUT OF MEMORY'
                        *
                        ********************
                        * OUTPUT TO RS232 *
                        ********************
                        *
                        * Transmits a character to the RS232 interface via
                        * the TICC if the interface is ready for it.
                        * In case of a car.ret, also a line feed is send.
                        *
                        * Entry: A: Character to be transmitted.
                        * Exit:  ABCDEHL preserved, F corrupted.
                        *
DD94 F5                 OUTSE   PUSH  PSW       Preserve char
DD95 3A00FD             OTS10   LDA   :FD00
DD98 E608                       ANI   :08       Check peripheral ready
DD9A CA95DD                     JZ    :DD95     Wait untill ready
DD9D 3AF3FF             OTS20   LDA   :FFF3
DDA0 E610                       ANI   :10       Check TICC buffer empty
DDA2 CA9DDD                     JZ    :DD9D     Wait untill empty
DDA5 F1                         POP   PSW       Get char
DDA6 32F6FF                     STA   :FFF6     Load serial output buffer
DDA9 FE0D                       CPI   :0D       Carriage return?
DDAB C0                         RNZ             Ready if not

                        * If car.ret:

DDAC F5                         PUSH  PSW
DDAD 3E0A                       MVI   A,:0A
DDAF CD94DD                     CALL  :DD94     Send line feed too
DDB2 F1                         POP   PSW
DDB3 C9                         RET
                        *
                        *********************
                        * INPUT FROM RS232 *
                        *********************
                        *
                        * Gets inputs from RS232 via TICC. Only 7-bit
                        * Ascii-code is accepted.
                        *
                        * Entry: No conditions.
                        * Exit:  A: Character received (0 if nothing).
                        *        BCDEHL preserved.
                        *
                        CINC
DDB4 3AF3FF             INSER   LDA   :FFF3
DDB7 E608                       ANI   :08       Check if something received
DDB9 C8                         RZ              Abort if no reception
DDBA 3AF0FF                     LDA   :FFF0     Received char in A
DDBD E67F                       ANI   :7F       Mask bit 7
DDBF C9                         RET
                        *
                        ***********************************
                        * RS232 FRAME ERROR - (not used) *
                        ***********************************
                        *
                        * Break test for serial input line.
```

```
                        *
DDC0 3AF3FF     BRSER   LDA   :FFF3
DDC3 1F                 RAR
DDC4 D0                 RNC             Abort if no break
DDC5 3AF3FF     BRS10   LDA   :FFF3     If break: check again
DDC8 1F                 RAR
DDC9 DAC5DD             JC    :DDC5     Wait until end of break
DDCC 3AF0FF             LDA   :FFF0     Load received character
DDCF 3F                 CMC             Set CY=1
DDD0 C9                 RET
                        *
                        *
                        *  ============================
                        *** ENCODING SERVICE ROUTINES ***
                        *  ============================
                        *
                        *
                        * The following routines are used both in 'main'
                        * and 'decode' modules.
                        *
                        ************************************************
                        * GET CHARACTER FROM LINE, NEGLECT TAB + SPACE *
                        ************************************************
                        *
                        * Entry: C: Position on current line.
                        * Exit:  Character in A; tab and space neglected.
                        *        C: Points to next character.
                        *        BDEHL preserved.
                        *
DDD1 0C         IGNBR   INR   C         Pnts to next char
DDD2 CDE0DD     IGNB    CALL  :DDE0     Get char from line
DDD5 FE20               CPI   :20       Space?
DDD7 CAD1DD             JZ    :DDD1     Then get next char
DDDA FE09               CPI   :09       Tab?
DDDC CAD1DD             JZ    :DDD1     Then get next char
DDDF C9                 RET
                        *
                        ****************************
                        * GET CHARACTER TO ENCODE *
                        ****************************
                        *
                        * Returns a character from some position on the
                        * current line. The source is determined by EFSW.
                        *
                        * Entry: C: Position on current line (max. 219).
                        * Exit:  EFSW=0  - Keyboard: Char on line pos in A.
                        *        EFSW>=2 - Edit buf: Char on EFEPT + line
                        *                            pos in A.
                        *        EFSW=1  - String:   Idem. If COUNT=line pos
                        *                            then char is car.ret.
                        *        F corrupted, BCDEHL preserved.
                        *
DDE0 3A3501     EFETCH  LDA   :0135
DDE3 FE01               CPI   :01       Check input direction
DDE5 DAFFDD             JC    :DDFF     If from keyb/RS232
DDE8 C2F4DD             JNZ   :DDF4     If from edit buffer

                        * If from string:

DDEB 3A3401             LDA   :0134     If string: Get COUNT
DDEE B9                 CMP   C         COUNT=pos.on curr.line?
DDEF 3E0D               MVI   A,:0D     Then char is car.ret
```



```
DDF1 CAFEDD             JZ    :DDFE     And abort

                        * Entry if from edit buffer:

DDF4 E5         EFC10   PUSH  H
DDF5 2A3201             LHLD  :0132     Get EFEPT
DDF8 79                 MOV   A,C
DDF9 CD30DE             CALL  :DE30     Add curr.line pos to EFEPT
DDFC 7E                 MOV   A,M       Get character
DDFD E1                 POP   H
DDFE C9         EFC20   RET

                        * If from screen:

DDFF EF         EFC30   RST   5         Get character from line
DE00 15                 DATA  :15
DE01 C9                 RET
                        *
                        *
                        *  ================================
                        *** SINGLE AND DOUBLE BYTE UTILITIES ***
                        *  ================================
                        *
                        *
                        ********************************
                        * CHECK IF UPPER CASE CHARACTER *
                        ********************************
                        *
                        * Entry: A: Character to be checked.
                        * Exit:  CY=0: Not upper case.
                        *        CY=1: Upper case.
                        *        ABCDEHL preserved, F corrupted.
                        *
DE02 FE41       ALPHA   CPI   :41       Lowest upper case char
DE04 3F                 CMC
DE05 D0                 RNC
DE06 FE5B               CPI   :5B       First lower case char
DE08 C9                 RET
                        *
                        ************************************************
                        * CHECK IF CHARACTER IS NUMBER OR UPPER CASE *
                        ************************************************
                        *
                        * Entry: A: Character to be checked.
                        * Exit:  CY=0: Not number, not upper case.
                        *        CY=1: Number or upper case.
                        *        ABCDEHL preserved. F corrupted.
                        *
DE09 CD02DE     ALNUM   CALL  :DE02     Check if upper case
DE0C D8                 RC
DE0D FE30       NUMER   CPI   :30       Lowest number
DE0F 3F                 CMC
DE10 D0                 RNC
DE11 FE3A               CPI   :3A       No number anymore
DE13 C9                 RET
                        *
                        **********************
                        * COMPARE HL AND DE *
                        **********************
                        *
                        * Compares HL with DE (HL-DE).
                        *
```

```
                        * Exit: DE=HL: Z=1, CY=0.
                        *       DE<HL: Z=0, CY=0.
                        *       DE>HL: Z=0, CY=1.
                        *       BCDEHL preserved, AF corrupted.
                        *
DE14 7C         COMP    MOV   A,H
DE15 BA                 CMP   D
DE16 C0                 RNZ
DE17 7D                 MOV   A,L
DE18 BB                 CMP   E
DE19 C9                 RET
                        *
                        ******************************
                        * CALCULATE LENGTH OF BLOCK *
                        ******************************
                        *
                        * Sets HL=HL-DE.
                        *
                        * Entry: Startaddress in DE, 1st address after
                        *        block in HL.
                        * Exit:  Length in HL, startaddress in DE.
                        *        If DE>HL, length in 2-complement.
                        *        ABCDE preserved, F as in COMP.
                        *
DE1A C5         SUBDE   PUSH  B
DE1B F5                 PUSH  PSW
DE1C 7D                 MOV   A,L
DE1D 93                 SUB   E         Calc. difference lowest byte
DE1E 6F                 MOV   L,A
DE1F 7C                 MOV   A,H
DE20 9A                 SBB   D         Calc. diff. highest byte
DE21 67                 MOV   H,A
DE22 C1                 POP   B
DE23 78                 MOV   A,B
DE24 C1                 POP   B
DE25 C9                 RET
                        *
                        ******************************
                        * DOUBLE BYTE TWO COMPLEMENT *
                        ******************************
                        *
                        * Sets HL=-HL.
                        *
                        * Entry: Double byte to be converted in HL.
                        * Exit:  Two complement in HL. ABCDEF preserved.
                        *
DE26 F5         CMPHL   PUSH  PSW
DE27 7C                 MOV   A,H
DE28 2F                 CMA             Complement H
DE29 67                 MOV   H,A
DE2A 7D                 MOV   A,L
DE2B 2F                 CMA             Complement L
DE2C 6F                 MOV   L,A
DE2D 23                 INX   H         Add 1
DE2E F1                 POP   PSW
DE2F C9                 RET
                        *
                        ***************************
                        * ADD OFF-SET TO ADDRESS *
                        ***************************
                        *
                        * Adds a given offset to a base address (HL=HL+A).
```



```
                        *
                        * Entry: Base in HL, offset in A.
                        * Exit:  HL=HL+A. ABCDE preserved, F corrupted.
                        *
DE30 F5         DADA    PUSH  PSW
DE31 85                 ADD   L
DE32 6F                 MOV   L,A       L=L+A
DE33 7C                 MOV   A,H
DE34 CE00               ACI   :00       Add carry if overflow
DE36 67                 MOV   H,A
DE37 F1                 POP   PSW
DE38 C9                 RET
                        *
                        ***********************************
                        * CALCULATE ADDRESS AFTER STRING *
                        ***********************************
                        *
                        * Sets HL=HL+M+1.
                        *
                        * Entry: HL points to 1st byte of string (length
                        *        byte).
                        * Exit:  HL points to first byte after string.
                        *        AFBCDE preserved.
                        *
DE39 F5         DADM    PUSH  PSW
DE3A 7E                 MOV   A,M       Get length of string
DE3B 23                 INX   H         HL: addr. 1st char. byte
DE3C CD30DE             CALL  :DE30     Calc addr after string
DE3F F1                 POP   PSW
DE40 C9                 RET
                        *
                        *****************
                        * DELAY ROUTINE *
                        *****************
                        *
                        * Runs a fixed delay loop of 665 msec. If
                        * interrupts are enabled, the delay will be
                        * approx. 750 msec.
                        * HL is loaded with FFFF, and then decremented.
                        *
                        * Exit: ABCDEHL preserved; F corrupted.
                        *
DE41 E5         DELAY   PUSH  H
DE42 D5                 PUSH  D
DE43 21FFFF             LXI   H,:FFFF   Init. delay value
DE46 54                 MOV   D,H
DE47 5D                 MOV   E,L
DE48 19         DLY10   DAD   D
DE49 DA48DE             JC    :DE48     Repeat if not ready
DE4C D1                 POP   D
DE4D E1                 POP   H
DE4E C9                 RET
                        *
                        ************************
                        * DATA BLOCK TRANSFER *
                        ************************
                        *
                        * Moves a block of data starting at (DE) and
                        * ending at (HL)-1 to (BC).
                        *
                        * Entry: DE: Startaddr. source bank.
                        *        BC: Startaddr. destination bank.
```

```
                        *       HL: Points after end source bank.
                        * Exit:  AF preserved, BCDEHL corrupted.
                        *
DE4F F5         MOVE    PUSH    PSW
DE50 E5                 PUSH    H
DE51 CD1ADE             CALL    :DE1A       Calc. length source bank
DE54 79                 MOV     A,C
DE55 93                 SUB     E
DE56 78                 MOV     A,B
DE57 9A                 SBB     D
DE58 DA6CDE             JC      :DE6C       If destination addr. is
                                            lower than source addr.

                        * Destination address > source address:

DE5B 54                 MOV     D,H
DE5C 5D                 MOV     E,L         Save length in DE
DE5D 09                 DAD     B           Highest dest.addr. in HL
DE5E C1                 POP     B
DE5F 7A         MOV10   MOV     A,D         Check if ready
DE60 B3                 ORA     E
DE61 CA7ADE             JZ      :DE7A       Then abort
DE64 1B                 DCX     D
DE65 2B                 DCX     H
DE66 0B                 DCX     B
DE67 0A                 LDAX    B           Get byte to be transferred
DE68 77                 MOV     M,A         Transfer it
DE69 C35FDE             JMP     :DE5F       Next one

                        * Destination address < source address:

DE6C 7C         MOV20   MOV     A,H
DE6D B5                 ORA     L
DE6E CA79DE             JZ      :DE79       Abort if ready
DE71 2B                 DCX     H
DE72 1A                 LDAX    D           Get byte to be transferred
DE73 02                 STAX    B           Transfer it
DE74 13                 INX     D
DE75 03                 INX     B
DE76 C36CDE             JMP     :DE6C       Next byte

                        * If ready:

DE79 E1         MOV30   POP     H
DE7A F1         MOV40   POP     PSW
DE7B C9                 RET
                        *
                        ********************************
                        * FILL BANK WITH IDENTICAL DATA *
                        ********************************
                        *
                        * Fills an area of memory with a constant.
                        *
                        * Entry: DE: Startaddr. of bank.
                        *        HL: Points after bank.
                        *        A:  Data to be loaded into bank.
                        * Exit:  DE: Points after bank.
                        *        BCHL preserved, AF corrupted.
                        *
DE7C C5         FILL    PUSH    B
DE7D 47                 MOV     B,A         Save data in B
DE7E CD14DE     FIL10   CALL    :DE14       Check if bank full
```



```
DE81 CA8DDE             JZ      :DE8D       Abort if ready
DE84 DA8DDE             JC      :DE8D       Abort if DE>HL
DE87 78                 MOV     A,B         Get data
DE88 12                 STAX    D           and store it
DE89 13                 INX     D
DE8A C37EDE             JMP     :DE7E       Next addr.
DE8D C1         FIL20   POP     B
DE8E C9                 RET
                        *
                        ********************
                        * MULTIPLY HL BY A *
                        ********************
                        *
                        * Multiplies a 16-bit value by a 8-bit value.
                        *
                        * Entry: HL: 16-bit value.
                        *        A:  8-bit value.
                        * Exit:  CY=0: Result in HL.
                        *        CY=1: Overflow.
                        *        ABCDE preserved.
                        *
DE8F 37         HLMUL   STC
DE90 F5                 PUSH    PSW
DE91 D5                 PUSH    D
DE92 EB                 XCHG
DE93 210000             LXI     H,:0000     Init. result
DE96 B7         HLM10   ORA     A
DE97 1F                 RAR                 Next bit of multiplier
DE98 D29FDE             JNC     :DE9F       Jump if bit is 0
DE9B 19                 DAD     D           Add 1* HL if bit is 1
DE9C DAADDE             JC      :DEAD       Abort if overflow
DE9F B7         HLM20   ORA     A
DEA0 CAB1DE             JZ      :DEB1       Abort if ready
DEA3 EB                 XCHG
DEA4 29                 DAD     H           Multiply *2
DEA5 EB                 XCHG
DEA6 D296DE             JNC     :DE96       Again if no overflow
DEA9 00                 NOP
DEAA 00                 NOP
DEAB 00                 NOP
DEAC 00                 NOP
DEAD 00         HLM90   NOP
DEAE D1                 POP     D
DEAF F1                 POP     PSW         Error exit if overflow
DEB0 C9                 RET
DEB1 D1         HLM99   POP     D
DEB2 F1                 POP     PSW
DEB3 3F                 CMC                 No error exit
DEB4 C9                 RET
                        *
                        *
                        *
DEB5                    END
```

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

ALNUM   DE09    ALPHA   DE02    BRS10   DDC5    BRSER   DDC0
CINC    DDB4    CMPHL   DE26    COLO    DD55    COMP    DE14
COUTC   DD6A    CRLF    DD5E    DADA    DE30    DADM    DE39
DELAY   DE41    DLY10   DE48    EFC10   DDF4    EFC20   DDFE
```

EFC30	DDFF	EFETCH	DDE0	EXIT1	DD45	FIL10	DE7E
FIL20	DE8D	FILL	DE7C	HLM10	DE96	HLM20	DE9F
HLM90	DEAD	HLM99	DEB1	HLMUL	DE8F	IGNB	DDD2
IGNBR	DDD1	INPLO	DD1A	INPLN	DD1F	INSER	DDB4
IPL10	DD22	IPL20	DD2A	IPL30	DD49	MOV10	DE5F
MOV20	DE6C	MOV30	DE79	MOV40	DE7A	MOVE	DE4F
NUMER	DE0D	OTBIN	DD75	OTC10	DD70	OTC20	DD8E
OTC99	DD8C	OTS10	DD95	OTS20	DD9D	OUTC	DD60
OUTSE	DD94	SCCHR	DD60	SUBDE	DE1A		


```
002                             ORG   :DEB5
003                     *
004                     *
005                     *
006                     *  ==================================
007                     *** BASIC EXECUTION / RUN-TIME MODULE ***
008                     *  ==================================
009                     *
010                     * Generally, BC is used as entry pointer to the
011                     * textbuffer.
012                     *
013                     ********************
014                     * RUN basiccmd NEW *
015                     ********************
016                     *
017                     * Sets up heap, empty textbuffer and symboltable,
018                     * sets pointers of textbuffer and symboltable
019                     * correctly. Old buffer contents is not destroyed,
020                     * (except 4 locations), but not useable.
021                     * Valid as direct command only.
022                     *
023                     * Exit: A=1, CY=1.
024                     *
025 DEB5 00             RNEW    NOP
026 DEB6 00                     NOP
027 DEB7 00                     NOP
028 DEB8 2A9B02                 LHLD  :029B     Get startaddr heap
029 DEBB 229F02                 SHLD  :029F     Set start textbuf=start heap
030 DEBE 3600                   MVI   M,:00     Store 00 in 1st addr.
031 DEC0 23                     INX   H
032 DEC1 22A102                 SHLD  :02A1     Set start symtab
033 DEC4 3600                   MVI   M,:00     00 in 1st addr.
034 DEC6 23                     INX   H
035 DEC7 22A302                 SHLD  :02A3     Set end symtab
036
037                     * Entry from scratch/edit:
038
039 DECA 2A9D02         HRINIT  LHLD  :029D     Get heap size
040 DECD EB                     XCHG            in DE
041 DECE CD95D1                 CALL  :D195     Init heap to all available
042 DED1 3E01                   MVI   A,:01     Code for 'buffer crunched'
043 DED3 37                     STC
044 DED4 C9                     RET
045                     *
046                     **********************
047                     * RUN basiccmd CONT *
048                     **********************
049                     *
050                     * Resets step flag, decr. CONFL (error if it was
051                     * 0), restores FRAME from stack, restores stack-
052                     * pointer and continues program execution.
053                     * Valid as direct command only.
054                     *
055 DED5 AF             RCONT   XRA   A
056 DED6 321601         RCN10   STA   :0116     Reset step flag
057 DED9 212601                 LXI   H,:0126   Get pntr suspended program
058 DEDC 35                     DCR   M         Update it
059 DEDD 3E19                   MVI   A,:19
060 DEDF FAF5D9                 JM    :D9F5     Evt. run error 'CAN'T CONT'
061 DEE2 2A2701                 LHLD  :0127     Get current base stack level
062 DEE5 EB                     XCHG            in DE
063 DEE6 211500                 LXI   H,:0015   FRAME length
```

```
064 DEE9 19             DAD   D          Top of FRAME in stack
065 DEEA E5             PUSH  H          Save it
066 DEEB 010001         LXI   B,:0100    Addr SYSBOT
067 DEEE CD4FDE         CALL  :DE4F      Load FRAME from stack
068 DEF1 E1             POP   H          Get addr FRAME top in stack
069 DEF2 222701         SHLD  :0127      Update current base stack
070                                      level
071 DEF5 F9             SPHL             Update stackpointer
072 DEF6 2A0201         LHLD  :0102      Get start current command
073 DEF9 44             MOV   B,H
074 DEFA 4D             MOV   C,L        Store it in BC
075 DEFB C37FC8         JMP   :C87F      Run BASIC line
076                     *
077                     **********************
078                     * RUN basiccmd STEP *
079                     **********************
080                     *
081 DEFE 3EFF           RSTEP MVI   A,:FF      Init value STEP flag
082 DF00 C3D6DE         JMP   :DED6      Set STEP flag and continu
083                     *
084                     **********************
085                     * RUN basiccmd STOP *
086                     **********************
087                     *
088                     * Entry: No conditions.
089                     * Exit:  A=03, CY=1. HL points after string.
090                     *
091 DF03 CDFFDA         RSTOP CALL  :DAFF      Print 'STOPPED'
092 DF06 C9DB           DBL   :DBC9
093 DF08 3E03           MVI   A,:03      Code for 'susp. execution'
094 DF0A 37             STC
095 DF0B C9             RET
096                     *
097                     **********************
098                     * RUN basiccmd END *
099                     **********************
100                     *
101                     * Entry: No conditions.
102                     * Exit:  A=01, CY=1. HL points after string.
103                     *
104 DF0C CDFFDA         REND  CALL  :DAFF      Print 'END PROGRAM'
105 DF0F CFDB           DBL   :DBCF
106 DF11 3E01           MVI   A,:01      Code for 'stop execution'
107 DF13 37             STC
108 DF14 C9             RET
109                     *
110                     ********************
111                     * RUN basiccmd IF *
112                     ********************
113                     *
114                     * Used for IF .. GOTO <linenr> and for
115                     * IF .. THEN <linenr>.
116                     *
117                     RIFG
118 DF15 CD63E7         RIFTL CALL  :E763      (0) Run logical expression
119 DF18 3C             INR   A
120 DF19 CA63DF         JZ    :DF63      Run GOTO if condition true
121                     
122                     * If condition false:
123                     
124 DF1C 03                   INX   B
125 DF1D 03                   INX   B          Skip linenr
```



```
126 DF1E B7                   ORA   A          No special action
127 DF1F C9                   RET
128                     *
129                     ****************************************
130                     * RUN basiccmd IF .. THEN <statement> *
131                     ****************************************
132                     *
133 DF20 CD63E7         RIFTC CALL  :E763      (0) Run logical expression
134 DF23 3C                   INR   A
135 DF24 C28FE1               JNZ   :E18F      (0) Ignore rest if false
136                     
137                     * If condition true:
138                     
139 DF27 03                   INX   B          Skip length line
140 DF28 B7                   ORA   A          No special action
141 DF29 C9                   RET              Execute rest of line
142                     *
143                     ********************
144                     * basiccmd GOSUB *
145                     ********************
146                     *
147                     * Saves current program state on the interpreter
148                     * stack and branches to a named line. The running
149                     * FOR loop (if any) is saved in order to avoid
150                     * problems if any unpaired NEXT is encountered.
151                     * The stackpointer at the subroutine-entry is held
152                     * to enable breaking out of a FOR-NEXT loop.
153                     *
154 DF2A CDEDE6         RGOSUB CALL  :E6ED      (0) Get linenr and find it
155                     
156                     * Entry from ON - GOSUB:
157                     
158 DF2D D1               RGS10  POP   D          Kill return addr
159 DF2E 221901                  SHLD  :0119      Save new PC
160 DF31 CD3CE1               CALL  :E13C      (0) Save FOR loop contents
161 DF34 C5                   PUSH  B          Save program position
162 DF35 2A1901               LHLD  :0119      Get new PC
163 DF38 44                   MOV   B,H        ) Is new text position
164 DF39 4D                   MOV   C,L        )
165 DF3A 2A1301               LHLD  :0113      Get stack level last GOSUB
166 DF3D E5                   PUSH  H          Save evt link to previous
167                                            subroutine entry
168 DF3E 210000               LXI   H,:0000
169 DF41 220401               SHLD  :0104      No running loop
170 DF44 39                   DAD   SP         SP in HL
171 DF45 221301               SHLD  :0113      SP in STKGOS for return
172                     
173                     * Entry on return:
174                     
175 DF48 B7             RGS20 ORA   A          No special action
176 DF49 C38FC8               JMP   :C88F      Into Basic monitor
177                     *
178                     **********************
179                     * RUN basiccmd RETURN *
180                     **********************
181                     *
182                     * Reverses the effect of a previous 'GOSUB'.
183                     *
184 DF4C D1             RRET  POP   D          Kill returnaddr
185 DF4D 2A1301               LHLD  :0113      Get stack level last GOSUB
186 DF50 7C                   MOV   A,H
187 DF51 B5                   ORA   L          0 if no active call
```

```
188 DF52 3E01              MVI   A,:01
189 DF54 CAF5D9            JZ    :D9F5      Then run error 'RETURN
190                                         WITHOUT GOSUB'
191 DF57 F9                SPHL             Else: re-instate old stack
192 DF58 E1                POP   H
193 DF59 221301            SHLD  :0113      Link back to previous
194                                         GOSUB
195 DF5C C1                POP   B          Restore text pntr
196 DF5D CD67E1            CALL  :E167      (0) Pop FRAME
197 DF60 C348DF            JMP   :DF48      Back to Basic monitor
198                *
199                **********************
200                * RUN basiccmd GOTO *
201                **********************
202                *
203                * Simply transfers control to a named line.
204                *
205                * Entry from IF - GOTO:
206                *
207 DF63 CDEDE6    RGOTO   CALL  :E6ED      (0) Get linenr and find it
208
209                * Entry from ON - GOTO:
210
211 DF66 44          RGT10   MOV   B,H        ) Set textpointer to line
212 DF67 4D                  MOV   C,L        )
213 DF68 B7                  ORA   A          No special action
214 DF69 C9                  RET
215                *
216                ****************************
217                * RUN basiccmd ON .. GOTO *
218                ****************************
219                *
220 DF6A CD78DF    RONGT   CALL  :DF78      Process command
221 DF6D DA66DF            JC    :DF66      Use GOTO if OK
222 DF70 C9                RET              If outside list
223                *
224                ****************************
225                * RUN basiccmd ON .. GOSUB *
226                ****************************
227                *
228 DF71 CD78DF    RONGS   CALL  :DF78      Process command
229 DF74 DA2DDF            JC    :DF2D      Use GOSUB if OK
230 DF77 C9                RET              If outside list
231                *
232                ************************
233                * COMMON 'ON ..' CODE *
234                ************************
235                *
236                * Exit: CY=1: OK.
237                *       CY=0: Outside list.
238                *
239 DF78 CD1DE7    RONFN   CALL  :E71D      ) (0) Get index of number
240 DF7B 5F                 MOV   E,A        ) in list in E
241 DF7C 0A                LDAX  B          Get nr of linenrs in list
242 DF7D 03                INX   B
243 DF7E 6F                MOV   L,A
244 DF7F 2600              MVI   H,:00
245 DF81 29                DAD   H
246 DF82 09                DAD   B          Pointer after list
247 DF83 E5                PUSH  H          Save pointer
248 DF84 1D                DCR   E
249 DF85 1C                INR   E
```



```
250 DF86 CA9BDF            JZ    :DF9B      Index of 0: outside list
251 DF89 BB                CMP   E
252 DF8A DA9BDF            JC    :DF9B      If index too large
253 DF8D 1600              MVI   D,:00
254 DF8F 1B                DCX   D
255 DF90 EB                XCHG
256 DF91 29                DAD   H
257 DF92 09                DAD   B          Pntr to reqd linenr
258 DF93 44                MOV   B,H        ) in BC
259 DF94 4D                MOV   C,L        )
260 DF95 CDEDE6            CALL  :E6ED      (0) Find reqd line
261 DF98 C1                POP   B
262 DF99 37                STC              CY=1: OK
263 DF9A C9                RET
264
265                * If outside list:
266
267 DF9B C1        ROF10   POP   B
268 DF9C B7        ROF11   ORA   A          CY=0: outside list
269 DF9D C9                RET
270                *
271                **********************
272                * RUN basiccmd RUN *
273                **********************
274                *
275                * No linenumber is given.
276                *
277 DF9E 210000    RRUN    LXI   H,:0000
278 DFA1 221501            SHLD  :0115      Reset trace/step flag
279 DFA4 2A9F02            LHLD  :029F      Get start textbuf
280 DFA7 44        RRN10   MOV   B,H        )
281 DFA8 4D                MOV   C,L        ) Store it in BC
282 DFA9 CD01E4            CALL  :E401      (0) Run RESTORE; Set data
283                                         pntr to start program
284 DFAC 3100F9            LXI   SP,:F900   Reset stackpointer
285 DFAF AF                XRA   A
286 DFB0 322601            STA   :0126      No suspended program
287 DFB3 CD23CB            CALL  :CB23      Empty HEAP + symtab
288 DFB6 B7                ORA   A          No special action
289 DFB7 C38FC8            JMP   :C88F      Run program
290                *
291                ******************************
292                * RUN basiccmd RUN <linenr> *
293                ******************************
294                *
295                * After RUN, a line number is given.
296                *
297 DFBA CDEDE6    RRUNN   CALL  :E6ED      (0) Read linenr and find
298                                         it in textbuf
299 DFBD C3A7DF            JMP   :DFA7      Process RUN
300                *
301                **********************
302                * RUN basiccmd POKE *
303                **********************
304                *
305 DFC0 CDF8E6    RPOKE   CALL  :E6F8      (0) Get addr in HL
306
307                * Entry for other modules:
308
309 DFC3 CD1DE7    RPEN    CALL  :E71D      (0) Get argument in A
310 DFC6 77                MOV   M,A        Store it
311 DFC7 B7                ORA   A          No special action
```

```
312 DFC8 C9                 RET
313                 *
314                 ********************
315                 * RUN basiccmd OUT *
316                 ********************
317                 *
318 DFC9 CD1DE7      ROUT    CALL  :E71D     (0) Get portnr in A
319 DFCC 57                 MOV   D,A       and in D
320 DFCD CD1DE7             CALL  :E71D     (0) Get data in A
321 DFD0 5F                 MOV   E,A       and in E
322 DFD1 B7                 ORA   A
323 DFD2 C3C8D8             JMP   :D8C8     Output to DCE-bus
324                 *
325                 ********************
326                 * RUN basiccmd WAIT *
327                 ********************
328                 *
329                 * WAIT I,J,K reads the status of Real World port
330                 * I, EXOR's it with K and AND's it with J until
331                 * a result equal to J is obtained.
332                 *
333 DFD5 CD1DE7      RWAIT   CALL  :E71D     (0) Get portnr
334 DFD8 57                 MOV   D,A       in D
335 DFD9 CD1DE7             CALL  :E71D     (0) Get bits needed high
336 DFDC 67                 MOV   H,A       in H
337 DFDD 0A                 LDAX  B         Get next byte from text
338 DFDE 03                 INX   B
339 DFDF D6FF               SUI   :FF       Check if ony 2 arguments
340 DFE1 00                 NOP
341 DFE2 C4DACE             CNZ   :CEDA     If 3 arg: Get XOR mask
342 DFE5 6F                 MOV   L,A       in L
343 DFE6 CDE0D8     RWT20   CALL  :D8E0     Input from DCE-bus
344 DFE9 7B                 MOV   A,E       into A
345 DFEA AD                 XRA   L         XOR with mask
346 DFEB A4                 ANA   H         AND with bits needed high
347 DFEC BC                 CMP   H         Correct value reached?
348 DFED C8                 RZ              Then abort
349 DFEE CDA5D6             CALL  :D6A5     Check keyb for new inputs
350 DFF1 D2E6DF             JNC   :DFE6     Next DCE-input if no Break
351
352                 * If suspended:
353
354 DFF4 C312E0             JMP   :E012     (0) Quit: 'cmd broken in'
355                 *
356                 ************************
357                 * RUN basiccmd WAIT MEM *
358                 ************************
359                 *
360                 * As WAIT, but with I is a memory location.
361                 *
362 DFF7 CDF8E6      RWTEM   CALL  :E6F8     (0) Get memory addr in HL
363 DFFA CD1DE7             CALL  :E71D     (0) Get bit mask
364 DFFD 57                 MOV   D,A       in D
365 DFFE 0A                 LDAX  B         Get next byte from text
366 DFFF 03                 INX   B
367 E000 D6FF               SUI   :FF       Check if only 2 arguments
368 E002 00                 NOP
369 E003 C4DACE             CNZ   :CEDA     If 3 arg: Get XOR mask
370 E006 5F                 MOV   E,A       in E
371 E007 7B         RWM20   MOV   A,E       XOR mask in A
372 E008 AE                 XRA   M         XOR with memory
373 E009 A2                 ANA   D         AND with bit mask
```



```
374 E00A BA                 CMP   D         Correct value reached?
375 E00B C8                 RZ              Then abort
376 E00C CDA5D6             CALL  :D6A5     Check keyb for inputs
377 E00F D207E0             JNC   :E007     Cont if no Break pressed
378
379                 * If suspended:
380
381 E012 3E02       RWT30   MVI   A,:02     Code 'command broken in'
382 E014 37                 STC
383 E015 C9                 RET
384                 *
385                 *
386                 *
387 E016                    END

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

HRINIT DECA   RCN10  DED6   RCONT  DED5   REND   DF0C
RGOSUB DF2A   RGOTO  DF63   RGS10  DF2D   RGS20  DF48
RGT10  DF66   RIFG   DF15   RIFTC  DF20   RIFTL  DF15
RNEW   DEB5   ROF10  DF9B   ROF11  DF9C   RONFN  DF78
RONGS  DF71   RONGT  DF6A   ROUT   DFC9   RPEN   DFC3
RPOKE  DFC0   RRET   DF4C   RRN10  DFA7   RRUN   DF9E
RRUNN  DFBA   RSTEP  DEFE   RSTOP  DF03   RWAIT  DFD5
RWM20  E007   RWT20  DFE6   RWT30  E012   RWTEM  DFF7
```