```
002                             ORG   :E5BC
003                     *
004                     *
005                     *
006                     ****************************************
007                     * ENCODE VARIABLE OR ARRAY REFERENCE *
008                     ****************************************
009                     *
010                     * L3E85: Reference to a variable or an array with
011                     *        arguments.
012                     * L3E86: Reference to an array without arguments.
013                     *
014                     * Entry: D : Code: 00: Reference to a value (array
015                     *                      with arguments or variable).
016                     *                  FF: Array name without arguments.
017                     *        C : Next position in input.
018                     *        HL: 1st free position in EBUF.
019                     * Exit:  C,HL updated, AF preserved.
020                     *        DE: Offset to symbol table (to T/L byte).
021                     *        B : T/L byte of name.
022                     *
023 E5BC 1600           L3E85   MVI   D,:00
024 E5BE F5             EVR10   PUSH  PSW
025 E5BF D5                     PUSH  D
026 E5C0 CDD2DD                 CALL  :DDD2     Get 1st char from line,
027                                             neglect tab + space
028 E5C3 CD02DE                 CALL  :DE02     Check if char is upper case
029 E5C6 D20BDA                 JNC   :DA0B     Run 'SYNTAX ERROR' if not
030 E5C9 E5                     PUSH  H
031 E5CA F5                     PUSH  PSW       Save 1st char
032
033                     * Check if name is a BASIC command:
034
035 E5CB 1E02                   MVI   E,:02     Nr of info bytes -1
036 E5CD 21BFCB                 LXI   H,:CBBF   Addr command table
037 E5D0 CD34CA                 CALL  :CA34     Find instr in table. On
038                                             exit, HL points to string
039                                             or after it if not found
040 E5D3 7E                     MOV   A,M       )
041 E5D4 E63F                   ANI   :3F       ) Check if end table reached
042 E5D6 FE25                   CPI   :25       )
043 E5D8 C20BDA                 JNZ   :DA0B     Run 'SYNTAX ERROR' if name
044                                             is a command
045 E5DB B7                     ORA   A
046
047                     * Check if name is a BASIC function:
048
049 E5DC CDFDE6                 CALL  :E6FD     Find var.name in input
050 E5DF 21E6CF                 LXI   H,:CFE6   Addr function table
051 E5E2 CD5ACA                 CALL  :CA5A     Find function in table
052 E5E5 DA0BDA                 JC    :DA0B     Run 'SYNTAX ERROR' if name
053                                             is a function
054
055                     * Check type marker in input:
056
057 E5E8 0C                     INR   C         Points to next char in input
058 E5E9 1E02                   MVI   E,:02     2 bytes in symtab for STR
059 E5EB 2620                   MVI   H,:20     String type byte
060 E5ED FE24                   CPI   :24
061 E5EF CA0EE6                 JZ    :E60E     Jump if STR ('$')
062 E5F2 1E04                   MVI   E,:04     4 byte in symtab for INT/FPT
063 E5F4 2610                   MVI   H,:10     INT type byte
```

```
064 E5F6 FE25               CPI   :25
065 E5F8 CA0EE6             JZ    :E60E     Jump if INT ('%')
066 E5FB 2600               MVI   H,:00     FPT type byte
067 E5FD FE21               CPI   :21
068 E5FF CA0EE6             JZ    :E60E     Jump if FPT ('!')
069
070                 * If no type marker given:
071
072 E602 F1                 POP   PSW       Get 1st byte var.name
073 E603 213402             LXI   H,:0234   Baseaddr IMPTAB
074 E606 CD30DE             CALL  :DE30     Calc offset addr in HL
075 E609 66                 MOV   H,M       Get type marker in H
076 E60A 0D                 DCR   C
077 E60B C30FE6             JMP   :E60F
078
079                 * Handle type marker:
080
081 E60E F1         L3E87   POP   PSW       Get 1st byte of name
082 E60F 7C         L3E88   MOV   A,H       type in A
083 E610 B2                 ORA   D         OR type (high nibble) with
084                                         length (low nibble)
085 E611 57                 MOV   D,A       T/L on name in D
086 E612 E1                 POP   H
087 E613 F1                 POP   PSW       Get code 00/FF in A
088 E614 E5                 PUSH  H
089 E615 CD18E0             CALL  :E018     ) Update EBUF pointer
090 E618 CD18E0             CALL  :E018     ) 2 positions
091 E61B B7                 ORA   A         Flags on code
092 E61C CA26E6             JZ    :E626     Jump if value
093
094                 * If name:
095
096 E61F 7A                 MOV   A,D       Get T/L byte of name
097 E620 F640               ORI   :40       Set bit 6 (array)
098 E622 57                 MOV   D,A       Preserve it
099 E623 C32EE6             JMP   :E62E
100
101                 * If value:
102
103 E626 CDE0DD     L3E89   CALL  :DDE0     Get char from line
104 E629 FE28               CPI   :28       '(' ?
105 E62B CC53E6             CZ    :E653     Then encode arguments
106                 *
107 E62E E3         L3E90   XTHL
108 E62F E5                 PUSH  H
109 E630 7A                 MOV   A,D       Get T/L byte on name
110 E631 E630               ANI   :30       Type only
111 E633 323601             STA   :0136     Set type latest expression
112 E636 D5                 PUSH  D         Preserve T/L name
113 E637 CD57CA             CALL  :CA57     Find variable in symtab
114 E63A D1                 POP   D         Get T/L name
115 E63B D47DE6             CNC   :E67D     Insert variable in symtab
116                                         if it is a new one
117 E63E 42                 MOV   B,D       T/L name in B
118 E63F EB                 XCHG            Var.addr in symtab in DE
119 E640 2AA102             LHLD  :02A1     Get startaddr symtab
120 E643 EB                 XCHG            in DE; var.addr in HL
121 E644 CD1ADE             CALL  :DE1A     Calc offset from begin
122                                         symtab in HL
123 E647 EB                 XCHG            Offset in DE
124 E648 E1                 POP   H         Retrieve EBUF pntr
125 E649 7A                 MOV   A,D       Hibyte offset in A
```

```
126 E64A F640               ORI   :40       Set bit 6 (array)
127 E64C 77                 MOV   M,A       Hibyte offset in EBUF
128 E64D 23                 INX   H
129 E64E 73                 MOV   M,E       Lobyte offset in EBUF
130 E64F 23                 INX   H
131 E650 E1                 POP   H
132 E651 F1                 POP   PSW
133 E652 C9                 RET
134                 *
135                 * ENCODE ARRAY ARGUMENTS:
136                 *
137                 * An arguments list is encoded into EBUF.
138                 * Format:
139                 *       nr of arg / type of arg / code for expr/
140                 *       < type of arg / code for expr >.
141                 *
142                 * Entry: D : T/L byte of variable name.
143                 *        C : Points to '(' of argument list for
144                 *            array in input.
145                 *        HL: 1st free position EBUF.
146                 * Exit:  D : 'Subscripted' flag.
147                 *        E : Nr of bytes in symtab (02).
148                 *        C,HL updated. B preserved. A=D.
149                 *
150 E653 1E00       EVR50  MVI   E,:00      Parameter count
151 E655 E5                  PUSH  H
152 E656 CD18E0            CALL  :E018     Update EBUF pointer
153 E659 1C       EVR55  INR   E          Parameter count +1
154 E65A 0C                INR   C         Skip 'C' or ','
155 E65B D5                PUSH  D
156 E65C CDB2E3            CALL  :E3B2     Encode non-boolean expr
157                                        preceeded by its type
158 E65F D1                POP   D
159 E660 CDD2DD            CALL  :DDD2     Get char from line, neglect
160                                        tab + space
161 E663 FE2C              CPI   :2C
162 E665 CA59E6            JZ    :E659     Get next parameter if it
163                                        is ','
164 E668 FE29              CPI   :29
165 E66A C20BDA            JNZ   :DA0B     Run 'SYNTAX ERROR' if
166                                        not ')'
167 E66D 0C                INR   C         Skip ')'
168 E66E E3                XTHL            Get old EBUF pntr
169 E66F 73                MOV   M,E       Parameter count into EBUF
170 E670 E1                POP   H
171 E671 1E02              MVI   E,:02     2 bytes space in symtab
172 E673 7A                MOV   A,D
173 E674 F640              ORI   :40       Set type is array
174 E676 57                MOV   D,A       Set flag 'subscripted'
175 E677 C9                RET
176                 *
177                 ************************
178                 * ENCODE AN ARRAY NAME *
179                 ************************
180                 *
181 E678 16FF       EARRN  MVI   D,:FF     Code for name only
182 E67A C3BEE5            JMP   :E5BE     Encode array name
183                 *
184                 ****************************************
185                 * INSERT A NEW VARIABLE IN SYMBOL TABLE *
186                 ****************************************
187                 *
```

```
188                     * The variable name is inserted in the symbol table
189                     * and the value is cleared.
190                     *
191                     * Entry: See CAB8.
192                     * Exit:  HL: Points to 2nd T/L byte of entry.
193                     *         AF corrupted, BCDE preserved.
194                     *
195 E67D CDB8CA         EVARI   CALL  :CAB8     Insert var.name in symtab
196 E680 E5                     PUSH  H
197 E681 23                     INX   H         HL pnts after 2nd T/L byte
198                                             of entry
199 E682 7A                     MOV   A,D       Get T/L of name
200 E683 E640                   ANI   :40
201 E685 C295E6                 JNZ   :E695     Jump if array type
202
203                     * If number type:
204
205 E688 7A                     MOV   A,D       Get T/L of name
206 E689 E630                   ANI   :30
207 E68B FE20                   CPI   :20
208 E68D CA95E6                 JZ    :E695     Jump if string type
209 E690 CD9ECB                 CALL  :CB9E     Clear value in symtab
210 E693 E1                     POP   H
211 E694 C9                     RET
212
213                     * If string/array type:
214
215 E695 3600           EVI10   MVI   M,:00     ) Clear pointer in symbtab
216 E697 23                     INX   H         )
217 E698 3600                   MVI   M,:00     )
218 E69A E1                     POP   H
219 E69B C9                     RET
220                     *
221                     ******************************
222                     * STORE QUOTED TEXT IN EBUF *
223                     ******************************
224                     *
225 E69C 3618           L3E96   MVI   M,:18     Code for quoted string (#18)
226                                             into EBUF
227 E69E CD18E0                 CALL  :E018     Update EBUF pointer
228 E6A1 1EFF                   MVI   E,:FF     Text must end with '"'
229 E6A3 C3B5E6                 JMP   :E6B5     Into common end
230                     *
231                     ************************************
232                     * STORE UNQUOTED STRING IN EBUF *
233                     ************************************
234                     *
235 E6A6 1E01           L3E97   MVI   E,:01     Text must end with ','
236 E6A8 3619                   MVI   M,:19     Code for unquoted string
237                                             (#19) into EBUF
238 E6AA CD18E0                 CALL  :E018     Update EBUF pointer
239 E6AD C3B5E6                 JMP   :E6B5     Into common end
240                     *
241                     ************************
242                     * STORE TEXT INTO EBUF *
243                     ************************
244                     *
245                     * Text in DATA, REM and '***' statements is moved
246                     * into the EBUF.
247                     *
248 E6B0 CDD2DD         L3E98   CALL  :DDD2     Get char from line, neglect
249                                             tab + space
```

```
250 E6B3 1E02                MVI   E,:02     Text must end with CR
251                                           Into common end
252                  *
253                  ************************************
254                  * COMMON END TEXT ENCODING ROUTINES *
255                  ************************************
256                  *
257                  * Entry: C : Points to 1st actual character to be
258                  *            stored.
259                  *        HL: Points to place for length byte in EBUF
260                  *        E : Handling switch:
261                  *             > 1: (but <#80): Text must end with CR
262                  *             = 1: Text will end with ',' (',' is no
263                  *                  inserted into EBUF).
264                  *            <= 0: Text will end at '"' ('"' is not
265                  *                  inserted into the EBUF).
266                  * Exit:  C : Points beyond text in input.
267                  *        HL: Points beyond stored text in EBUF.
268                  *        D : Length of stored text.
269                  *        A : Character which marks end of text.
270                  *        B preserved, E corrupted.
271                  *
272 E6B5 E5        L3E99   PUSH  H
273 E6B6 CD18E0            CALL  :E018     Update EBUF pointer
274 E6B9 1600            MVI   D,:00     Set length is 0
275 E6BB CDE0DD    L3E100  CALL  :DDE0     Get char from line
276 E6BE FE0D            CPI   :0D
277 E6C0 CADBE6          JZ    :E6DB     Jump if char is 'CR'
278 E6C3 FE2C            CPI   :2C
279 E6C5 CAE3E6          JZ    :E6E3     Jump if char is ','
280 E6C8 0C        L3E101  INR   C
281 E6C9 FE22            CPI   :22
282 E6CB C2D3E6          JNZ   :E6D3     Jump if char is not '"'
283 E6CE 1D              DCR   E
284 E6CF FADFE6          JM    :E6DF     If done: store length in
285                                       EBUF, quit
286 E6D2 1C              INR   E
287
288                  * Character into EBUF:
289
290 E6D3 77        L3E102  MOV   M,A       Load char in EBUF
291 E6D4 CD18E0            CALL  :E018     Update EBUF pointer
292 E6D7 14              INR   D
293 E6D8 C3BBE6          JMP   :E6BB     Get next char
294
295                  * If 'CR':
296
297 E6DB 1D        L3E103  DCR   E
298 E6DC FA0BDA          JM    :DA0B     If E >= #80: Run 'SYNTAX
299                                       ERROR'
300 E6DF E3        L3E104  XTHL
301 E6E0 72              MOV   M,D       Length in EBUF entry
302 E6E1 E1              POP   H
303 E6E2 C9              RET
304
305                  * If ',':
306
307 E6E3 1D        L3E105  DCR   E
308 E6E4 CADFE6          JZ    :E6DF     If E=0: Store length in
309                                       EBUF, quit
310 E6E7 C3ABE8          JMP   :E8AB     incr E, get next char
311                  *
```

```
                        ***********************************************
                        * FIND BINARY OR UNITARY OPERATOR IN TABLE *
                        ***********************************************
                        *
                        * Entry/exit: See #3E6F6.
                        *
E6EA E5         L3E106  PUSH  H
E6EB 2191CF             LXI   H,:CF91   Startaddr table
E6EE 1E00       L3E107  MVI   E,:00
E6F0 CD34CA             CALL  :CA34     Find instr in table
E6F3 7E                 MOV   A,M       Get code from table
E6F4 E1                 POP   H
E6F5 C9                 RET
                        *
                        ***************************************
                        * FIND AN UNITARY OPERATOR IN TABLE *
                        ***************************************
                        *
                        * Routine looks for a init. string beginning at
                        * C in table.
                        *
                        * Entry: C : Points to input.
                        * Exit:  CY=0: Not found:
                        *              C : Points to 1st valid character
                        *                  after entry address.
                        *              A : Contains code info 0.
                        *              DE = 0, BHL preserved.
                        *        CY=1: Found:
                        *              C : Points beyond string found.
                        *              A : Code byte from table.
                        *              DE = 0, BHL preserved.
                        *
E6F6 E5         L3E108  PUSH  H
E6F7 21D8CF             LXI   H,:CFD8   Startaddr table
E6FA C3EEE6             JMP   :E6EE     Into previous routine
                        *
                        *
                        *
E6FD                    END
```

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

EARRN  E678   EVARI  E67D   EVI10  E695   EVR10  E5BE
EVR50  E653   EVR55  E659   L3E100 E6BB   L3E101 E6C8
L3E102 E6D3   L3E103 E6DB   L3E104 E6DF   L3E105 E6E3
L3E106 E6EA   L3E107 E6EE   L3E108 E6F6   L3E85  E5BC
L3E87  E60E   L3E88  E60F   L3E89  E626   L3E90  E62E
L3E96  E69C   L3E97  E6A6   L3E98  E6B0   L3E99  E6B5
```

```
002                             ORG   :E6FD
003                     *
004                     *
005                     *
006                     ********************************
007                     * FIND VARIABLE NAME IN INPUT *
008                     ********************************
009                     *
010                     * Checks if 1st character is a upper case one.
011                     * Reads the input (starting with character after
012                     * C) till it finds a non-alphanumeric character
013                     * (number or upper case). On a carry CALL, it also
014                     * accepts %, ! or $ at the end. Blanks are not
015                     * ignored and not accepted.
016                     *
017                     * Entry: C :  Input position.
018                     * Exit:  B :  Entry C.
019                     *        C :  Points to 1st character not accepted.
020                     *        D :  Count of 1st character not accepted.
021                     *             (1st character read has count 1).
022                     *        A,E: 1st non-alphanumeric character read.
023                     *        HL preserved, F corrupted.
024                     *
025 E6FD F5             RDID    PUSH  PSW       Preserve CY-flag
026 E6FE 41                     MOV   B,C
027 E6FF 1600                   MVI   D,:00     Init count
028 E701 14             L3E110  INR   D         Count +1
029 E702 0C                     INR   C         Line pos +1
030 E703 7A                     MOV   A,D       Count in A
031 E704 FE0F                   CPI   :0F       Max 14 char for a name
032 E706 DA0AE7                 JC    :E70A
033 E709 15                     DCR   D         Skip last char if > 14
034 E70A CDE0DD         L3E111  CALL  :DDE0     Get char from line
035 E70D CD09DE                 CALL  :DE09     Check if nr or upper case
036 E710 DA01E7                 JC    :E701     Get next char if O.K.
037 E713 CDE0DD                 CALL  :DDE0     Get 1st non-alphanum.char
038                                             from line
039 E716 5F                     MOV   E,A       Store it in E
040 E717 F1                     POP   PSW       Get CY-flag back
041 E718 7B                     MOV   A,E       Get char back in A
042 E719 D0                     RNC             Abort if non-carry CALL
043
044                     * On carry CALL only: accept !,%,$:
045
046 E71A FE25                   CPI   :25
047 E71C CA27E7                 JZ    :E727     Jump if char is '%'
048 E71F FE21                   CPI   :21
049 E721 CA27E7                 JZ    :E727     Jump if char is '!'
050 E724 FE24                   CPI   :24
051 E726 C0                     RNZ             Abort if char is not '$'
052 E727 0C             L3E112  INR   C         Update line pos
053 E728 14                     INR   D         Update count
054 E729 C9                     RET
055                     *
056                     ********************
057                     * GET LINE NUMBER *
058                     ********************
059                     *
060                     * Exit: C,HL: Updated.
061                     *       B preserved, AFDE corrupted.
062                     *       'SYNTAX ERROR' if no linenr given.
```

```
064 E72A CD31E7     ELN     CALL  :E731     Read linenr into EBUF
065 E72D D20BDA             JNC   :DA0B     Run 'SYNTAX ERROR' if no
066                                         number given
067 E730 C9                RET
068                 *
069                 ********************************
070                 * READ LINE NUMBER INTO EBUF *
071                 ********************************
072                 *
073                 * Exit: CY=0: No linenumber given. Error exit if
074                 *             linenumber is 0 or > #FFFF.
075                 *       CY=1: O.K.
076                 *       C,HL: Updated.
077                 *       B preserved, AFDE corrupted.
078                 *
079 E731 CDD2DD     L3E114  CALL  :DDD2     Get char from line, neglect
080                                         tab + space
081 E734 CD24C0             CALL  :C024     Input INT number to MACC
082 E737 D0                 RNC             Abort if no linenr given
083 E738 C5                 PUSH  B
084 E739 E7                 RST   4         Copy MACC into reg ABCD
085 E73A 15                 DATA  :15
086 E73B B0                 ORA   B
087 E73C C251E7             JNZ   :E751     Error exit if > #FFFF
088 E73F B1                 ORA   C
089 E740 B2                 ORA   D
090 E741 CA51E7             JZ    :E751     Error exit if nr = 0
091 E744 5A                 MOV   E,D       ) Linenr in DE
092 E745 51                 MOV   D,C       )
093 E746 72                 MOV   M,D       Hibyte linenr into EBUF
094 E747 CD18E0             CALL  :E018     Update EBUF pointer
095 E74A 73                 MOV   M,E       Lobyte linenr into EBUF
096 E74B CD18E0             CALL  :E018     Update EBUF pointer
097 E74E C1                 POP   B
098 E74F 37                 STC             CY=1 (O.K.)
099 E750 C9                 RET
100
101                 * Error exit:
102
103 E751 C1         L3E115  POP   B
104 E752 3E15               MVI   A,:15
105 E754 C3F5D9             JMP   :D9F5     Run error 'NUMBER OUT OF
106                                         RANGE'
107                 *
108                 ****************************************
109                 * ENCODE BINARY OPERATION INTO EBUF *
110                 ****************************************
111                 *
112                 * Entry: A : Result of 3E7CF:
113                 *            1xx xxxxx : compute type / opcode.
114                 *        E : Table code byte.
115                 *        HL: 1st free location EBUF.
116                 *
117 E757 F5         L3E116  PUSH  PSW
118 E758 7B                MOV   A,E       Get conversion code byte
119 E759 EB                XCHG
120 E75A 2A3B01            LHLD  :013B     Get addr last operator in
121                                         EBUF
122 E75D EB                XCHG            in DE
123 E75E CD70E7            CALL  :E770     Add conversion byte for
124                                         2nd operand
125 E761 0F                RRC
```

```
126 E762 0F                 RRC
127 E763 EB                 XCHG
128 E764 2A3901             LHLD  :0139     Get addr in EBUF for next
129                                         operator
130 E767 EB                 XCHG
131 E768 CD70E7             CALL  :E770     Add conversion byte for
132                                         1st operand
133 E76B F1                 POP   PSW       Restore compute/opcode in A
134 E76C CD83E7             CALL  :E783     Insert it into EBUF
135 E76F C9                 RET
136                *
137                ********************************************
138                * ADD INT/FPT CONVERSION BYTE TO EXPRESSION *
139                ********************************************
140                *
141                * Entry: A : Conversion byte:
142                *            #01: Convert FPT to INT.
143                *            #02: Convert INT to FPT.
144                *
145 E770 F5        L3E117  PUSH  PSW
146 E771 E603               ANI   :03       Conversion only
147 E773 CA81E7             JZ    :E781     Jump if no conversion reqd
148 E776 1F                 RAR             CY=1 if FPT to INT
149 E777 3E9F               MVI   A,:9F     Conv.byte FPT to INT
150 E779 DA7EE7             JC    :E77E
151 E77C 3EBF               MVI   A,:BF     Conv.byte INT to FPT
152 E77E CD83E7     L3E118  CALL  :E783     Insert conv.byte into EBUF
153 E781 F1         L3E119  POP   PSW
154 E782 C9                 RET
155                *
156                ***************************
157                * INSERT BYTE INTO EBUF *
158                ***************************
159                *
160                * Data is moved 1 byte to create space for byte
161                * to be inserted.
162                *
163                * Entry: A : Byte to be inserted.
164                *        DE: Startaddress source bank.
165                *        HL: Endaddress source bank +1.
166                * Exit:  HL= HL + 1.
167                *        ABCDE preserved.
168                *
169 E783 C5         L3E120  PUSH  B
170 E784 42                 MOV   B,D       ) Start source in BC
171 E785 4B                 MOV   C,E       )
172 E786 03                 INX   B         Destination 1 byte higher
173 E787 F5                 PUSH  PSW
174 E788 D5                 PUSH  D
175 E789 CD18E0             CALL  :E018     Update EBUF pointer
176 E78C E5                 PUSH  H
177 E78D 2B                 DCX   H
178 E78E CD4FDE             CALL  :DE4F     Move EBUF contents 1 byte
179 E791 E1                 POP   H
180 E792 D1                 POP   D
181 E793 F1                 POP   PSW
182 E794 12                 STAX  D         Store byte into EBUF
183 E795 C1                 POP   B
184 E796 C9                 RET
185                *
186                *
187                *
```

```
                        ****************************************
                        * ENCODE AN UNITARY OPERATOR FOR A TERM *
                        ****************************************
                        *
                        * Entry: A : Code according to table CFD8.
                        * Exit:  BCHL preserved. DE corrupted.
                        *        A : Code byte:
                        *                    +     -    INOT
                        *             INT    BC    BD    BE
                        *             FPT    9C    9D    *
                        *
E797 F5         L3E121  PUSH  PSW
E798 213601             LXI   H,:0136   Addr type last expression
E79B E61F               ANI   :1F       Opcode only
E79D FE00               CPI   :00       '+' ?
E79F 161C               MVI   D,:1C
E7A1 CABAE7             JZ    :E7BA     Then jump
E7A4 FE01               CPI   :01       '-' ?
E7A6 161D               MVI   D,:1D
E7A8 CABAE7             JZ    :E7BA     Then jump
E7AB FE1E               CPI   :1E       'INOT' ?
E7AD C20BDA             JNZ   :DA0B     Run 'SYNTAX ERROR' if not

                        * If 'INOT':

E7B0 7E                 MOV   A,M       Get type last expression
E7B1 FE10               CPI   :10       Must be INT
E7B3 C21ADA             JNZ   :DA1A     Run error 'TYPE MISMATCH'
                                        if not
E7B6 3EBE               MVI   A,:BE     Code 'INOT' in A
E7B8 E1         L3E122  POP   H
E7B9 C9                 RET

                        * If '+' or '-':

E7BA 7E         L3E123  MOV   A,M       Get type last expression
E7BB FE10               CPI   :10
E7BD 1EA0               MVI   E,:A0
E7BF CACAE7             JZ    :E7CA     Jump if INT
E7C2 7E                 MOV   A,M       Get type last expression
E7C3 FE00               CPI   :00
E7C5 1E80               MVI   E,:80
E7C7 C21ADA             JNZ   :DA1A     Run error 'TYPE MISMATCH'
                                        if not FPT
E7CA 7A         L3E124  MOV   A,D       ) Set up code in A
E7CB B3                 ORA   E         )
E7CC C3B8E7             JMP   :E7B8
                        *
                        ****************************************
                        * OBTAIN TYPE INFO FOR BINARY OPERATION *
                        ****************************************
                        *
                        * Entry: A : Code for binary operation (lower 5
                        *            bits).
                        *        E : Type 1st operand.
                        *     TYPE: Type 2nd operand.
                        *
                        * Routine compares both types. If different, one
                        * must be INT and the other FPT, else type mismatch
                        * error.
                        * Type conversion and operation type are obtained
                        * from table on 3E835. Type mismatch if illegal
```

```
250                 * type.
251                 * Type of result is stored in TYPE.
252                 *
253                 * Exit: E : Type code from table.
254                 *       A : Code for EBUF:  1xx xxxxx:
255                 *           bits 5,6: type of compute required:
256                 *                     0 : FPT
257                 *                     1 : INT
258                 *                     2 : STR
259                 *                     3 : Boolean
260                 *           bits 0-4: Opcode.
261                 *       BCDHL preserved.
262                 *
263 E7CF E5          L3E125  PUSH  H
264 E7D0 D5                  PUSH  D
265 E7D1 E61F                ANI   :1F       Opcode only
266 E7D3 F5                  PUSH  PSW
267 E7D4 CD1FE8              CALL  :E81F     Set D according to opcode
268 E7D7 3A3601              LDA   :0136     Get type latest expression
269 E7DA BB                  CMP   E         Compare both types
270 E7DB C209E8              JNZ   :E809     Jump if not identical
271 E7DE 07                  RLC             )
272 E7DF 07                  RLC             ) Type code from TYPE in
273 E7E0 07                  RLC             ) lonibble
274 E7E1 07                  RLC             )
275 E7E2 E603                ANI   :03       Only lower 2 bits
276 E7E4 6F                  MOV   L,A       in L
277 E7E5 7A          L3E126  MOV   A,D       Get opcode group (0-5)
278 E7E6 87                  ADD   A         *2
279 E7E7 57                  MOV   D,A       in D
280 E7E8 87                  ADD   A         *4
281 E7E9 82                  ADD   D         *6 (find group in table)
282 E7EA 85                  ADD   L         Find pos in grouptable
283 E7EB 2135E8              LXI   H,:E835   Startaddr result table
284 E7EE CD30DE              CALL  :DE30     Find addr resultcode in
285                                          table
286 E7F1 7E                  MOV   A,M       Get resultcode
287 E7F2 3C                  INR   A         Check if code is FF
288 E7F3 CA1ADA              JZ    :DA1A     Then run error 'TYPE
289                                          MISMATCH'
290 E7F6 3D                  DCR   A
291 E7F7 E630                ANI   :30       Get type of result only
292 E7F9 323601              STA   :0136     Store type latest expression
293 E7FC D1                  POP   D         Get code for binary
294                                          operation in D
295 E7FD 5E                  MOV   E,M       ) Get resultcode from table
296 E7FE 7E                  MOV   A,M       ) in E and in A
297 E7FF 1F                  RAR
298 E800 E660                ANI   :60       Reqd computing in bits 5,6
299 E802 B2                  ORA   D         Add opcode in bits 0-4
300 E803 F680                ORI   :80       Set bit 7
301 E805 E1                  POP   H
302 E806 54                  MOV   D,H       Restore D
303 E807 E1                  POP   H
304 E808 C9                  RET
305
306                 * If both types not identical:
307
308 E809 2E04        L3E127  MVI   L,:04
309 E80B FE00                CPI   :00       TYPE is FPT ?
310 E80D CA16E8              JZ    :E816     Then jump
311 E810 2C                  INR   L
```

```
312 E811 FE10                   CPI   :10       TYPE is INT ?
313 E813 C21ADA                 JNZ   :DA1A     Run error 'TYPE MISMATCH'
314                                             if not
315 E816 83             L3E128  ADD   E         Add other type
316 E817 FE10                   CPI   :10       Result must be #10
317 E819 C21ADA                 JNZ   :DA1A     Run error 'TYPE MISMATCH'
318                                             if not
319 E81C C3E5E7                 JMP   :E7E5     Calc conversion
320                     *
321                     * SET D DEPENDING ON OPCODE BINARY OPERATOR:
322                     *
323                     * Entry: A : Opcode binary operator (table #CF91).
324                     * Exit: ABCEHL preserved.
325                     *
326 E81F 1600            L3E129  MVI   D,:00     D=0
327 E821 FE01                   CPI   :01
328 E823 D8                     RC              Ready if opcode is 0 (+)
329 E824 14                     INR   D         D=1
330 E825 FE04                   CPI   :04
331 E827 D8                     RC              Ready if opcode is 1,2 or
332                                             3 (-,/,*)
333 E828 1602                   MVI   D,:02     D=2
334 E82A C8                     RZ              Ready if opcode is 4 (^)
335 E82B 14                     INR   D         D=3
336 E82C FE10                   CPI   :10
337 E82E D8                     RC              Ready if opcode is 5-F
                                                (IOR,IAND,IXOR,SHL,SHR,MOD)
338
339 E82F 14                     INR   D         D=4
340 E830 FE18                   CPI   :18
341 E832 D8                     RC              Ready if opcode is 10-17
                                                (>=, <=, >, <, =, <>)
342
343 E833 14                     INR   D         D=5
344 E834 C9                     RET             If opcode >= 18 (AND,OR)
345                     *
346                     * TABLE WITH TYPE RESULTS:
347                     *
348                     * The table gives the relation between input
349                     * operands, the binary operator and the result
350                     * for different groups of binary operations.
351                     * The groupnumber is calculated in 3E81F.
352                     *
353                     * Format each group: 6 bytes. Sequence:
354                     *             FPT/FPT
355                     *             INT/INT
356                     *             STR/STR
357                     *             LOGIC/LOGIC
358                     *             INT/FPT
359                     *             FPT/INT
360                     * Format each byte:
361                     *       bit 7,6: type arithmetic ) 0: FPT  1: INT
362                     *       bit 5,4: Type result     ) 2: STR  3: logic
363                     *       bit 3,2: Conversion left operand.
364                     *       bit 1,0: Conversion right operand.
365                     *                0 : No conversion.
366                     *                1 : Convert to INT.
367                     *                2 : Convert to FPT.
368                     *       FF : Not possible.
369                     *
370 E835 00              L3E385  DATA  :00       Group D=0:
371 E836 50                     DATA  :50       +
372 E837 A0                     DATA  :A0
373 E838 FF                     DATA  :FF
```

```
374 E839 08                     DATA  :08
375 E83A 02                     DATA  :02
376                   *
377 E83B 00                     DATA  :00       Group D=1:
378 E83C 50                     DATA  :50       -,/,*
379 E83D FF                     DATA  :FF
380 E83E FF                     DATA  :FF
381 E83F 08                     DATA  :08
382 E840 02                     DATA  :02
383                   *
384 E841 00                     DATA  :00       Group D=2:
385 E842 0A                     DATA  :0A       ^
386 E843 FF                     DATA  :FF
387 E844 FF                     DATA  :FF
388 E845 08                     DATA  :08
389 E846 02                     DATA  :02
390                   *
391 E847 55                     DATA  :55       Group D=3:
392 E848 50                     DATA  :50       IAND,IOR,IXOR,MOD,SHL,SHR
393 E849 FF                     DATA  :FF
394 E84A FF                     DATA  :FF
395 E84B 51                     DATA  :51
396 E84C 54                     DATA  :54
397                   *
398 E84D 30                     DATA  :30       Group D=4:
399 E84E 70                     DATA  :70       <,>,<>,=,<=,>=
400 E84F B0                     DATA  :B0
401 E850 FF                     DATA  :FF
402 E851 38                     DATA  :38
403 E852 32                     DATA  :32
404                   *
405 E853 FF                     DATA  :FF       Group D=5:
406 E854 FF                     DATA  :FF       AND,OR
407 E855 FF                     DATA  :FF
408 E856 F0                     DATA  :F0
409 E857 FF                     DATA  :FF
410 E858 FF                     DATA  :FF
411                   *
412                   *
413                   *
414 E859                        END
```

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

ELN     E72A    L3E110 E701    L3E111 E70A    L3E112 E727
L3E114 E731    L3E115 E751    L3E116 E757    L3E117 E770
L3E118 E77E    L3E119 E781    L3E120 E783    L3E121 E797
L3E122 E7B8    L3E123 E7BA    L3E124 E7CA    L3E125 E7CF
L3E126 E7E5    L3E127 E809    L3E128 E816    L3E129 E81F
L3E385 E835    RDID   E6FD
```

```
002                             ORG   :E859
003                     *
004                     *
005                     *
006                     ********************************
007                     * CHECK STATEMENT TERMINATOR *
008                     ********************************
009                     *
010                     * Get character from line and checks if it is
011                     * a correct terminator (':' or car.ret).
012                     *
013                     * Exit: Z=1: correct terminator.
014                     *       Z=0: incorrect.
015                     *       BCDEHL preserved. A corrupted.
016                     *
017 E859 CDD2DD         TSEOC   CALL  :DDD2     Get char from line, neglect
018                                             tab + space
019 E85C FE3A                   CPI   :3A       Is it ':' ?
020 E85E C8                     RZ
021 E85F FE0D                   CPI   :0D       Is it 'CR' ?
022 E861 C9                     RET
023                     *
024                     ************************************
025                     * CHECK IF NEXT CHARACTER IS ',' *
026                     ************************************
027                     *
028                     * Exit: C updated, AF corrupted, BDEHL preserved.
029                     *
030 E862 CD67E8         L3E131  CALL  :E867     Check if next char is ','
031 E865 2C                     DATA  :2C
032 E866 C9                     RET
033                     *
034                     ************************
035                     * CHECK NEXT CHARACTER *
036                     ************************
037                     *
038                     * Routine finds next valid character in input. If
039                     * it is not the character expected: syntax error.
040                     *
041                     * Entry: C : Points to input.
042                     *        ASCII-value of character to compare with
043                     *        on stack.
044                     * Exit:   If correct: C updated, AF corrupted,
045                     *        BDEHL preserved.
046                     *
047 E867 E3             ECHRI   XTHL            HL pnts to expected char
048 E868 CDD2DD                 CALL  :DDD2     Get char from line, neglect
049                                             tab + space
050 E86B BE                     CMP   M         Is it expected one ?
051 E86C C20BDA                 JNZ   :DA0B     Run 'SYNTAX ERROR' if not
052 E86F 0C                     INR   C         Pnts to next input
053 E870 23                     INX   H         )
054 E871 E3                     XTHL            ) Update SP
055 E872 C9                     RET
056                     *
057                     ********************************
058                     * ENCODE 'ERASE' - (not used) *
059                     ********************************
060                     *
061                     * The BASIC command 'ERASE' is cancelled.
062                     *
063 E873 1178E6         L3E133  LXI   D,:E678   Addr routine encode array
```

```
                                                  without arguments
E876 C32AE1             JMP   :E12A       Encode
                *
                *********************************
                * INPUT FPT NUMBER INTO MACC *
                *********************************
                *
                * Entry: Z=1: Change sign too.
                * Exit:  C updated, ABDEHL preserved.
                *         CY=0: Error.
                *
E879 CD1EC0     L3E134  CALL  :C01E       Input FPT number to MACC
E87C C0                 RNZ               Ready if Z=0
E87D E7                 RST   4           Else change sign MACC
E87E 1B                 DATA  :1B
E87F C9                 RET
                *
                *********************************
                * STORE QUOTED TEXT INTO EBUF *
                *********************************
                *
                * Entry: C points to 1st '"'.
                *
E880 0C         L3E135  INR   C
E881 C39CE6             JMP   :E69C       Store text in EBUF
                *
                *********************************
                * STORE A HEX NUMBER INTO EBUF *
                *********************************
                *
                * Entry: C points to '#' of hex number.
                *
E884 0C         EHEX    INR   C
E885 C390E5             JMP   :E590       Hex nr into EBUF
                *
                *************************************
                * ENCODE AN INT NUMBER INTO EBUF *
                *************************************
                *
E888 CDAFE8     L3E137  CALL  :E8AF       #14 into EBUF
E88B FE23               CPI   :23         '#' ?
E88D CA99E8             JZ    :E899       Then jump
E890 CD7BE5             CALL  :E57B       INT nr into EBUF
E893 D2B7E8     L3E139  JNC   :E8B7       Evt run 'SYNTAX ERROR'
E896 C35CE1             JMP   :E15C       Quit

                * If hex number:

E899 CD84E8     L3E138  CALL  :E884       Hex nr into EBUF
E89C C393E8             JMP   :E893
                *
E89F FF                 DATA  :FF
E8A0 FF                 DATA  :FF
E8A1 FF                 DATA  :FF
                *
                *****************
                * ENCODE 'DATA' *
                *****************
                *
E8A2 7D         EDATA   MOV   A,L
E8A3 FE42               CPI   :42
E8A5 C20BDA             JNZ   :DA0B       Run 'SYNTAX ERROR' if not
```

```
126 E8A8 C366E3              JMP   :E366     Encode text
127                      *
128                      ***********************************
129                      * part of END ENCODING (3E6B5) *
130                      ***********************************
131                      *
132 E8AB 1C              L3E140  INR   E
133 E8AC C3C8E6                 JMP   :E6C8
134                      *
135                      ************************************
136                      * CODE FOR INT NUMBER INTO EBUF *
137                      ************************************
138                      *
139                      * Gets also next character to encode.
140                      *
141 E8AF 3614            L3E141  MVI   M,:14     INT code (#14) in EBUF
142 E8B1 CD18E0                 CALL  :E018     Update EBUF pointer
143 E8B4 C3D2DD                 JMP   :DDD2     Get char from line, neglect
144                                             tab + space
145                      *
146                      ********************************************************
147                      * ERROR EXIT OF ENCODE INT NR INTO EBUF (3E888) *
148                      ********************************************************
149                      *
150 E8B7 2A3201          L3E142  LHLD  :0132     Get EFEPT
151 E8BA 11FCFF                 LXI   D,:FFFC
152 E8BD 19                     DAD   D         Set back linepntr
153 E8BE 220001                 SHLD  :0100     Store start current line
154 E8C1 C30BDA                 JMP   :DA0B     Run 'SYNTAX ERROR'
155                      *
156 E8C4 FF                     DATA  :FF
157                      *
158                      ****************************************
159                      * ASCII TABLE UPPER CASE (UNSHIFTED) *
160                      ****************************************
161                      *
162 E8C5 30              KEYTU   DATA  :30       0
163 E8C6 31                      DATA  :31       1
164 E8C7 32                      DATA  :32       2
165 E8C8 33                      DATA  :33       3
166 E8C9 34                      DATA  :34       4
167 E8CA 35                      DATA  :35       5
168 E8CB 36                      DATA  :36       6
169 E8CC 37                      DATA  :37       7
170 E8CD 38                      DATA  :38       8
171 E8CE 39                      DATA  :39       9
172 E8CF 3A                      DATA  :3A       :
173 E8D0 3B                      DATA  :3B       ;
174 E8D1 2C                      DATA  :2C       ,
175 E8D2 2D                      DATA  :2D       -
176 E8D3 2E                      DATA  :2E       .
177 E8D4 2F                      DATA  :2F       /
178 E8D5 0D                      DATA  :0D       car ret
179 E8D6 41                      DATA  :41       A
180 E8D7 42                      DATA  :42       B
181 E8D8 43                      DATA  :43       C
182 E8D9 44                      DATA  :44       D
183 E8DA 45                      DATA  :45       E
184 E8DB 46                      DATA  :46       F
185 E8DC 47                      DATA  :47       G
186 E8DD 48                      DATA  :48       H
187 E8DE 49                      DATA  :49       I
```

```
188 E8DF 4A                     DATA  :4A       J
189 E8E0 4B                     DATA  :4B       K
190 E8E1 4C                     DATA  :4C       L
191 E8E2 4D                     DATA  :4D       M
192 E8E3 4E                     DATA  :4E       N
193 E8E4 4F                     DATA  :4F       O
194 E8E5 50                     DATA  :50       P
195 E8E6 51                     DATA  :51       Q
196 E8E7 52                     DATA  :52       R
197 E8E8 53                     DATA  :53       S
198 E8E9 54                     DATA  :54       T
199 E8EA 55                     DATA  :55       U
200 E8EB 56                     DATA  :56       V
201 E8EC 57                     DATA  :57       W
202 E8ED 58                     DATA  :58       X
203 E8EE 59                     DATA  :59       Y
204 E8EF 5A                     DATA  :5A       Z
205 E8F0 5B                     DATA  :5B       [
206 E8F1 5E                     DATA  :5E       ^
207 E8F2 20                     DATA  :20       space
208 E8F3 00                     DATA  :00       (rept)
209 E8F4 08                     DATA  :08       char del
210 E8F5 10                     DATA  :10       cursor up
211 E8F6 11                     DATA  :11       cursor down
212 E8F7 12                     DATA  :12       cursor left
213 E8F8 13                     DATA  :13       cursor right
214 E8F9 09                     DATA  :09       tab
215 E8FA 80                     DATA  :80       ctrl
216 E8FB 00                     DATA  :00       (break)
217 E8FC 00                     DATA  :00       (shift)
218                     *
219                     ***************************************
220                     * ASCII TABLE LOWER CASE (SHIFTED) *
221                     ***************************************
222                     *
223 E8FD 30             KEYTS   DATA  :30       0
224 E8FE 21                     DATA  :21       !
225 E8FF 22                     DATA  :22       "
226 E900 23                     DATA  :23       #
227 E901 24                     DATA  :24       $
228 E902 25                     DATA  :25       %
229 E903 26                     DATA  :26       &
230 E904 27                     DATA  :27       '
231 E905 28                     DATA  :28       (
232 E906 29                     DATA  :29       )
233 E907 2A                     DATA  :2A       *
234 E908 2B                     DATA  :2B       +
235 E909 3C                     DATA  :3C       <
236 E90A 3D                     DATA  :3D       =
237 E90B 3E                     DATA  :3E       >
238 E90C 3F                     DATA  :3F       ?
239 E90D 0D                     DATA  :0D       car ret
240 E90E 61                     DATA  :61       a
241 E90F 62                     DATA  :62       b
242 E910 63                     DATA  :63       c
243 E911 64                     DATA  :64       d
244 E912 65                     DATA  :65       e
245 E913 66                     DATA  :66       f
246 E914 67                     DATA  :67       g
247 E915 68                     DATA  :68       h
248 E916 69                     DATA  :69       i
249 E917 6A                     DATA  :6A       j
```

```
250 E918 6B                 DATA  :6B       k
251 E919 6C                 DATA  :6C       l
252 E91A 6D                 DATA  :6D       m
253 E91B 6E                 DATA  :6E       n
254 E91C 6F                 DATA  :6F       o
255 E91D 70                 DATA  :70       p
256 E91E 71                 DATA  :71       q
257 E91F 72                 DATA  :72       r
258 E920 73                 DATA  :73       s
259 E921 74                 DATA  :74       t
260 E922 75                 DATA  :75       u
261 E923 76                 DATA  :76       v
262 E924 77                 DATA  :77       w
263 E925 78                 DATA  :78       x
264 E926 79                 DATA  :79       y
265 E927 7A                 DATA  :7A       z
266 E928 5D                 DATA  :5D       ]
267 E929 7E                 DATA  :7E       ~
268 E92A 20                 DATA  :20       space
269 E92B 00                 DATA  :00       (rept)
270 E92C 08                 DATA  :08       char del
271 E92D 14                 DATA  :14       window up
272 E92E 15                 DATA  :15       window down
273 E92F 16                 DATA  :16       window left
274 E930 17                 DATA  :17       window right
275 E931 09                 DATA  :09       tab
276 E932 80                 DATA  :80       ctrl
277 E933 00                 DATA  :00       (break)
278 E934 00                 DATA  :00       (shift)
279                     *
280                     ***************************************
281                     * GET INPUTS FROM KEYBOARD OR DINC *
282                     ***************************************
283                     *
284                     * Part of RESET (C719). Determines input source
285                     * depending on 1st input done.
286                     *
287 E935 CDBBD6     L3E143  CALL  :D6BB     Scan keyb; char in A
288 E938 D8                 RC              Ready if break pressed
289 E939 C0                 RNZ             Ready if key input done
290 E93A C3F4EF             JMP   :EFF4     Else: Get input from DINC
291                     *
292 E93D FF                 DATA  :FF
293 E93E FF                 DATA  :FF
294                     *
295                     ************************************************
296                     * LOAD ASCII VALUE FOR KEY PRESSED IN BUFFER *
297                     ************************************************
298                     *
299                     * From the key pressed, the offset to the start-
300                     * address of the ASCII table is calculated. The
301                     * ASCII value for the pressed key is stored in
302                     * the circular buffer KLIND.
303                     *
304                     * Entry: B : Column number.
305                     *        C : Row number.
306                     * Exit:  All registers preserved.
307                     *
308 E93F F5          INKEY   PUSH  PSW
309 E940 C5                 PUSH  B
310 E941 E5                 PUSH  H
311 E942 3E07               MVI   A,:07     )
```

```
312 E944 90                  SUB   B          )
313 E945 87                  ADD   A          ) Calc offset of startaddr
314 E946 87                  ADD   A          ) for key pressed.
315 E947 87                  ADD   A          )   Store it in C
316 E948 81                  ADD   C          )
317 E949 4F                  MOV   C,A        )
318 E94A 2AA702              LHLD  :02A7      Get startaddr ASCII table
319 E94D 0600                MVI   B,:00
320 E94F FE11                CPI   :11        )
321 E951 DA5DE9              JC    :E95D      ) Check if key is a char
322 E954 FE2B                CPI   :2B        ) A-Z
323 E956 D25DE9              JNC   :E95D      )
324 E959 3AC302              LDA   :02C3      Get shift lock value
325 E95C 47                  MOV   B,A        in B
326 E95D 3AB002      L3E145  LDA   :02B0      Get 'shift' byte
327 E960 A8                  XRA   B          Take CTRL into account
328 E961 E640                ANI   :40        A=#40 if shift, 00 when not
329 E963 CA6BE9              JZ    :E96B      Jump if no shift
330 E966 D5                  PUSH  D
331 E967 113800              LXI   D,:0038    Add. offset for lower case
332                                           table
333 E96A 19                  DAD   D          Startaddr lower case table
334                                           now in HL
335 E96B D1                  POP   D
336 E96C 0600        L3E146  MVI   B,:00
337 E96E 09                  DAD   B          Add offset to startaddr
338 E96F 7E                  MOV   A,M        Get ASCII value from table
339 E970 B7                  ORA   A          Check if Break, Rept, Shift
340 E971 CA8EE9              JZ    :E98E      Then Pop, ret
341 E974 FE80                CPI   :80        Check if CTRL
342 E976 CA92E9              JZ    :E992      Then update CTRL flag
343 E979 47                  MOV   B,A        Store ASCII value in B
344 E97A 2ABE02              LHLD  :02BE      Get addr next pos in KLIND
345 E97D E5                  PUSH  H          Store KLIIN on stack
346 E97E CD9CD6              CALL  :D69C      Update KLIND pointer
347 E981 3AC002              LDA   :02C0      Get lsbyte next output pos
348                                           of KLIND
349 E984 BD                  CMP   L          Compare with KLIIN
350 E985 CA8DE9              JZ    :E98D      Abort if buffer full
351 E988 22BE02              SHLD  :02BE      Update KLIIN
352 E98B E3                  XTHL             Get old KLIIN from stack
353 E98C 70                  MOV   M,B        Store ASCII char in KLIND
354 E98D E1          L3E147  POP   H
355 E98E E1          L3E148  POP   H
356 E98F C1                  POP   B
357 E990 F1                  POP   PSW
358 E991 C9                  RET
359
360                  * Update CTRL flag:
361
362 E992 3AC302      L3E149  LDA   :02C3      Get shiftlock value
363 E995 2F                  CMA              Invert it
364 E996 32C302              STA   :02C3      And store it again
365 E999 C38EE9              JMP   :E98E      Pop, ret
366                  *
367                  ****************
368                  * HEAP REQUEST *
369                  ****************
370                  *
371                  * The routine checks the heap for free areas. Evt.
372                  * consecutive free areas are consolidated. If this
373                  * procedure finds a free area min. 2 bytes larger
```

```
374                     * than requested, then it is reserved by setting
375                     * the length bytes (msb=0). An evt. resting free
376                     * area is set with length bytes and msb=1 (this
377                     * area must be >= 2 bytes).
378                     * The heap contents is never moved to obtain one
379                     * large consolidated area of free bytes!!
380                     *
381                     * Entry: DE: Length requested heap space.
382                     * Exit:  AFBCDE preserved.
383                     *        HL: Points to a 2-byte length of the re-
384                     *            quested gap. If no space available,
385                     *            it points to an error routine.
386                     *
387 E99C F5            HREQ    PUSH  PSW
388 E99D C5                    PUSH  B
389 E99E D5                    PUSH  D
390 E99F 42                    MOV   B,D       ) Reqd length in BC
391 E9A0 4B                    MOV   C,E       )
392 E9A1 2A9B02              HRQ10 LHLD  :029B     Get startaddr Heap
393 E9A4 56                    MOV   D,M       )
394 E9A5 23                    INX   H         ) Contents 1st 2 bytes of
395 E9A6 5E                    MOV   E,M       ) Heap in DE (length)
396 E9A7 23                    INX   H         )
397 E9A8 7A                    MOV   A,D       1st byte in A
398 E9A9 E67F                  ANI   :7F       Mask bit 'free/used'
399 E9AB BA                    CMP   D
400 E9AC 57                    MOV   D,A       D is length without msb
401 E9AD CA27D2                JZ    :D227     If area not free: check if
402                                              end of heap reached. JMP
403                                              #E9FA if not
404
405                     * If free area found: Check all next heap entries
406                     * and accumulate all free areas in succession:
407
408 E9B0 E5            HRQ20   PUSH  H         Save startaddr area +2
409 E9B1 19                    DAD   D         Begin next area in HL
410 E9B2 7E                    MOV   A,M
411 E9B3 B7                    ORA   A         Check msb of this area
412 E9B4 F2C7E9                JP    :E9C7     Jump if area occupied
413 E9B7 23                    INX   H
414 E9B8 7B                    MOV   A,E
415 E9B9 86                    ADD   M         Add lobyte length next area
416                                              length previous area
417 E9BA 5F                    MOV   E,A
418 E9BB 7A                    MOV   A,D
419 E9BC 2B                    DCX   H
420 E9BD 8E                    ADC   M         Add hibyte length next area
421                                              length previous area
422 E9BE E67F                  ANI   :7F       Skip bit 'free/occupied'
423 E9C0 57                    MOV   D,A
424 E9C1 13                    INX   D
425 E9C2 13                    INX   D         Add 2 extra bytes
426 E9C3 E1                    POP   H         Restore start free area +2
427 E9C4 C3B0E9                JMP   :E9B0     Check next area
428
429                     * Next area is not free:
430
431 E9C7 E1            HRQ30   POP   H         Restore start free area +2
432 E9C8 E5                    PUSH  H
433 E9C9 2B                    DCX   H
434 E9CA 73                    MOV   M,E       )
435 E9CB 2B                    DCX   H         ) Store total free length in
```

```
436 E9CC 7A                 MOV   A,D       ) 1st 2 bytes of free area
437 E9CD F680               ORI   :80       )
438 E9CF 77                 MOV   M,A       )
439                                         Space available here: HL pnt
440                                         to start free area +2; DE is
441                                         size free area; BC size reqd
442 E9D0 7B                 MOV   A,E       )
443 E9D1 91                 SUB   C         )
444 E9D2 6F                 MOV   L,A       ) Calc free length - reqd.
445 E9D3 7A                 MOV   A,D       ) length; result in HL
446 E9D4 98                 SBB   B         )
447 E9D5 67                 MOV   H,A       )
448 E9D6 FAF9E9             JM    :E9F9     Space not sufficient: Leave
449                                         consolidated area as free
450 E9D9 C2E4E9             JNZ   :E9E4     Jump if sufficient space
451 E9DC B5                 ORA   L
452 E9DD CAF0E9             JZ    :E9F0     Jump if just enough
453 E9E0 3D                 DCR   A
454 E9E1 CAF9E9             JZ    :E9F9     Not useable if 1 free byte
455                                         left
456
457                 * Set not used part of free area to free:
458
459 E9E4 EB          HR040  XCHG             Addr free area in DE
460 E9E5 1B                 DCX   D
461 E9E6 1B                 DCX   D         Reserve 2 bytes for length
462 E9E7 E1                 POP   H         Restore start free area +2
463 E9E8 E5                 PUSH  H
464 E9E9 09                 DAD   B         Add reqd length to find
465                                         start of resting free area
466 E9EA 7A                 MOV   A,D
467 E9EB F680               ORI   :80       Set free flag
468 E9ED 77                 MOV   M,A       )
469 E9EE 23                 INX   H         ) Length free area into heap
470 E9EF 73                 MOV   M,E       )
471
472                 * Reserve area for requested entry:
473
474 E9F0 E1          HR050  POP   H         Restore start free area +2
475 E9F1 2B                 DCX   H
476 E9F2 71                 MOV   M,C       )
477 E9F3 2B                 DCX   H         ) Reqd length in 1st 2 bytes
478 E9F4 70                 MOV   M,B       )
479 E9F5 D1                 POP   D
480 E9F6 C1                 POP   B
481 E9F7 F1                 POP   PSW
482 E9F8 C9                 RET
483
484                 * Area too small:
485
486 E9F9 E1          HR070  POP   H         Restore start free area
487 E9FA 19          HR075  DAD   D         HL pnts to next area
488 E9FB C3A4E9             JMP   :E9A4     Check next area
489                 *
490 E9FE FF                 DATA  :FF
491 E9FF FF                 DATA  :FF
492                 *
493                 *
494                 *
495                 *
496 EA00                    END
```

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

ECHRI	E867	EDATA	E8A2	EHEX	E884	HREQ	E99C
HRQ10	E9A4	HRQ20	E9B0	HRQ30	E9C7	HRQ40	E9E4
HRQ50	E9F0	HRQ70	E9F9	HRQ75	E9FA	INKEY	E93F
KEYTS	E8FD	KEYTU	E8C5	L3E131	E862	L3E133	E873
L3E134	E879	L3E135	E880	L3E137	E888	L3E138	E899
L3E139	E893	L3E140	E8AB	L3E141	E8AF	L3E142	E8B7
L3E143	E935	L3E145	E95D	L3E146	E96C	L3E147	E98D
L3E148	E98E	L3E149	E992	TSEOC	E859		