```
002                             ORG   :EA00
003                     *
004                     *
005                     *
006                     *  ================
007                     *** UTILITY PACKAGE ***
008                     *  ================
009                     *
010                     **************
011                     * (not used) *
012                     **************
013                     *
014 EA00 E1             L3E158  POP   H
015 EA01 C309EA                 JMP   :EA09
016                     *
017                     ***********************
018                     * RETURN AFTER 'GO' *
019                     ***********************
020                     *
021 EA04 E5             L3E159  PUSH  H           Returnaddr after 'GO'
022 EA05 21DCBA                 LXI   H,:BADC     Dummy 'saved PC' to prevent
023                                               continuation 'G' with start
024                                               address
025 EA08 E3                     XTHL              Returnaddr in HL
026 EA09 225900         L3E160  SHLD  :0059       Save HL
027 EA0C E1                     POP   H           in HL: #BADC
028                                               Into initialisation
029                     *
030                     ************************
031                     * INITIALISE UTILITY *
032                     ************************
033                     *
034                     * CPU registers are saved in the utility work
035                     * area. Input from the keyboard is awaited.
036                     *
037                     * The address EA42 is the general return address
038                     * for all Utility commands.
039                     *
040 EA0D 225D00         CALRX   SHLD  :005D       Save PC (next instr)
041 EA10 F5                     PUSH  PSW
042 EA11 E1                     POP   H
043 EA12 225300                 SHLD  :0053       Save PSW
044 EA15 210000                 LXI   H,:0000
045 EA18 39                     DAD   SP
046 EA19 225B00                 SHLD  :005B       Save SP
047 EA1C EB                     XCHG
048 EA1D 223900                 SHLD  0057        Save DE
049 EA20 60                     MOV   H,B
050 EA21 69                     MOV   L,C
051 EA22 225500                 SHLD  :0055       Save BC
052 EA25 CDE7ED                 CALL  :EDE7       Invert nibbles in CPU
053                                               reg save area
054 EA28 2A5D00                 LHLD  :005D       Get addr next instr
055 EA2B 7C                     MOV   A,H
056 EA2C B5                     ORA   L
057 EA2D CA3CEA                 JZ    :EA3C       If it is 0000 (entry from
058                                               BASIC)
059                                               Else:
060 EA30 2B                     DCX   H           HL on addr current instr
061 EA31 7E                     MOV   A,M         Get opcode in A
062 EA32 E6C7                   ANI   :C7
063 EA34 FEC7                   CPI   :C7
```

```
064 EA36 C23CEA              JNZ   :EA3C     Jump if instr is a RST
065 EA39 225D00              SHLD  :005D     Save addr next instr
066 EA3C 217DEA      L3E162  LXI   H,:EA7D   Startaddr string table
067 EA3F CD2FED              CALL  :ED2F     Print 'PC UTILITY V3.3'
068
069                  * UT command look-up:
070
071 EA42 CD3AED      L3E163  CALL  :ED3A     Print car.ret
072 EA45 0E3E                MVI   C,:3E
073 EA47 CDB4EE              CALL  :EEB4     Print '>'
074 EA4A CD06ED              CALL  :ED06     Get keyb input, print char
075 EA4D 2142EA              LXI   H,:EA42   Returnaddr in HL
076 EA50 E5                  PUSH  H         Save it on stack
077 EA51 218DEA              LXI   H,:EA8D   Startaddr table commands
078 EA54 23          L3E164  INX   H
079 EA55 BE                  CMP   M         Compare input with table
080 EA56 DA62EA              JC    :EA62     Error if invalid input
081 EA59 23                  INX   H         )
082 EA5A 5E                  MOV   E,M       ) Get pointer to address
083 EA5B 23                  INX   H         ) of part. routine in DE
084 EA5C 56                  MOV   D,M       )
085 EA5D C254EA              JNZ   :EA54     Check with next command
086 EA60 EB                  XCHG            Startaddr routine in HL
087 EA61 E9                  PCHL            Go to this routine
088                  *
089                  *********
090                  * ERROR *
091                  *********
092                  *
093                  * The only error message in utility is '?'.
094                  *
095                  * Exit: B preserved, AFCDEHL corrupted.
096                  *
097 EA62 CD3AED      ERROR   CALL  :ED3A     Print car.ret
098 EA65 0E3F                MVI   C,:3F
099 EA67 CDB4EE              CALL  :EEB4     Print '?'
100 EA6A 2A5B00      ERRST   LHLD  :005B     Get saved SP
101 EA6D F9                  SPHL            Restore stackpointer
102 EA6E 00                  NOP
103 EA6F 00                  NOP
104 EA70 00                  NOP
105 EA71 C342EA              JMP   :EA42     Start again for new input
106                  *
107                  **********************
108                  * ENTRY FROM BASIC *
109                  **********************
110                  *
111 EA74 225900      RESET   SHLD  :0059     Save HL
112 EA77 210000              LXI   H,:0000   Addr next instr (dummy)
113 EA7A C30DEA              JMP   :EA0D     Init utility
114                  *
115                  ***************************
116                  * UTILITY SCREEN HEADER *
117                  ***************************
118                  *
119                  * 0C is clear screen; 20 is space.
120                  *
121 EA7D 0C          MSGS0   DATA  :0C
122 EA7E 50                  DATA  :50       P
123 EA7F 43                  DATA  :43       C
124 EA80 20                  DATA  :20
125 EA81 55                  DATA  :55       U
```

```
126 EA82 54                     DATA  :54       T
127 EA83 49                     DATA  :49       I
128 EA84 4C                     DATA  :4C       L
129 EA85 49                     DATA  :49       I
130 EA86 54                     DATA  :54       T
131 EA87 59                     DATA  :59       Y
132 EA88 20                     DATA  :20
133 EA89 56                     DATA  :56       V
134 EA8A 33                     DATA  :33       3
135 EA8B 2E                     DATA  :2E       .
136 EA8C 33                     DATA  :33       3
137                     *
138 EA8D 00                     DATA  :00       End table
139                     *
140                     ********************************
141                     * TABLE WITH UTILITY COMMANDS *
142                     ********************************
143                     *
144 EA8E 42             CMDTB  DATA  :42       B
145 EA8F A0C7                  DBL   :C7A0     return addr to Basic
146                     *
147 EA91 44                     DATA  :44       D
148 EA92 B3EA                   DBL   :EAB3     startaddr Display
149                     *
150 EA94 46                     DATA  :46       F
151 EA95 48ED                   DBL   :ED48     startaddr Fill
152                     *
153 EA97 47                     DATA  :47       G
154 EA98 8AED                   DBL   :ED8A     startaddr Go
155                     *
156 EA9A 4C                     DATA  :4C       L
157 EA9B 26EB                   DBL   :EB26     startaddr Look
158                     *
159 EA9D 4D                     DATA  :4D       M
160 EA9E 83EC                   DBL   :EC83     startaddr Move
161                     *
162 EAA0 52                     DATA  :52       R
163 EAA1 0FEF                   DBL   :EF0F     startaddr Read
164                     *
165 EAA3 53                     DATA  :53       S
166 EAA4 5CED                   DBL   :ED5C     startaddr Substitute
167                     *
168 EAA6 56                     DATA  :56       V
169 EAA7 77ED                   DBL   :ED77     startaddr Vector Examine
170                     *
171 EAA9 57                     DATA  :57       W
172 EAAA E4EE                   DBL   :EEE4     startaddr Write
173                     *
174 EAAC 58                     DATA  :58       X
175 EAAD 6EED                   DBL   :ED6E     startaddr Examine
176                     *
177 EAAF 5A                     DATA  :5A       Z
178 EAB0 BAEC                   DBL   :ECBA     startaddr Reset
179                     *
180 EAB2 FF                     DATA  :FF       End table
181                     *
182                     ***************
183                     * D - DISPLAY *
184                     ***************
185                     *
186                     * Displays contents of memory. Two address values
187                     * are required which specify the range of memory
```

```
                        * to be displayed. Break will abort the print out.
                        *
                        * Exit: CY=1, All registers corrupted.
                        *
EAB3 0E02               DISPK   MVI   C,:02      Nr of addr inputs required
EAB5 CDDEEA                     CALL  :EADE      Get laddr and haddr on stack
EAB8 0D                         DCR   C
EAB9 F262EA                     JP    :EA62      Error if only 1 addr given
EABC D1                         POP   D          Haddr in DE
EABD E1                         POP   H          Laddr in HL
                        *
                        * Direct call entry:
                        *
                        DISL1
EABE CD3AED             DISP    CALL  :ED3A      Print car.ret
EAC1 CD18ED                     CALL  :ED18      Print laddr in ASCII
EAC4 CD01ED             DISL2   CALL  :ED01      Print space
EAC7 7E                         MOV   A,M        Get contents laddr
EAC8 CD1DED                     CALL  :ED1D      Print it in ASCII
EACB CD80ED                     CALL  :ED80      INX H; check if ready
EACE D8                         RC               Quit if ready
EACF CDC2EE                     CALL  :EEC2      Scan for Break pressed,
                                                 abort if pressed.
EAD2 7D                         MOV   A,L
EAD3 E60F                       ANI   :0F        Last instr on line?
EAD5 CABEEA                     JZ    :EABE      Then car.ret and continue
EAD8 C3C4EA                     JMP   :EAC4      Next addr
                        *
                        ***************************
                        * ADDRESS ARGUMENT INPUT *
                        ***************************
                        *
                        * The keyboard is scanned and the inputs are
                        * evaluated.
                        * (C) address arguments will be input and put on
                        * stack (LIFO). On return, C contains the
                        * difference between number entered and number
                        * desired. At least one argument is returned.
                        * Only the last 4 hex characters are used for the
                        * address value. Arguments are delimited by a
                        * space. The entry is terminated by CR, ESC, last
                        * argument or an invalid character (then error
                        * exit). Escape returns with CY set.
                        *
                        * Entry: C: max. nr of datablocks/addresses.
                        * Exit:  B: Last character typed in (terminator).
                        *        AFHL corrupted, DE preserved.
                        *
EADB AF                 ADALT   XRA   A          A=0
EADC B9                         CMP   C          Max nr of inputs reached?
EADD C8                         RZ               Then return
EADE 210000             ADARG   LXI   H,:0000
EAE1 CD06ED             ADACL   CALL  :ED06      Scan keyb, print char
EAE4 47                         MOV   B,A        Store char in B
EAE5 CD15EB                     CALL  :EB15      Convert it to hex
EAE8 DAF4EA                     JC    :EAF4      If char not 0-F: check if
                                                 delimiter
EAEB 29                 ADACE   DAD   H          )
EAEC 29                         DAD   H          ) Move bits 1 nibble
EAED 29                         DAD   H          ) (compose addr from inputs)
EAEE 29                         DAD   H          )
EAEF 85                         ADD   L          Add hex char to L
```

```
EAF0 6F              MOV   L,A       and store it
EAF1 C3E1EA          JMP   :EAE1     Next char

                * Check for delimiter/terminator:

EAF4 E3         ADADC   XTHL         Save given addr on stack
EAF5 E5                 PUSH  H      Save returnaddr again
EAF6 0D                 DCR   C      decr input counter
EAF7 78                 MOV   A,B    Get char last input
EAF8 FE20               CPI   :20    Space ?
EAFA CADBEA             JZ    :EADB  Then get next addr
EAFD FE0D               CPI   :0D    Car.ret ?
EAFF C8                 RZ           Then ready
EB00 FE12               CPI   :12    Escape?
EB02 37                 STC          Then CY=1
EB03 C8                 RZ           and return
EB04 C362EA             JMP   :EA62  Else goto Error

                * As ADARG, but carry return if 1st character is
                * not a legal hex digit:

EB07 210000     ADART   LXI   H,:0000  Immediate delimiter
EB0A CD06ED             CALL  :ED06    Scan keyb, print char
EB0D 47                 MOV   B,A      Store char in B
EB0E CD15EB             CALL  :EB15    Convert it to hex
EB11 D8                 RC             Return if no hex char
EB12 C3EBEA             JMP   :EAEB    Into previous routine
                *
                * CONVERT NUMBER FROM ASCII TO HEX-VALUE:
                *
                * Entry: Character in A (bit 7 must be 0).
                * Exit:   CY=0: Hex-value in A.
                *         CY=1: Input was not 0-F; (A) useless.
                *         BCDEHL preserved.
                *
EB15 D630       ASHEX   SUI   :30
EB17 D8                 RC             Error if #00-#2F
EB18 FE0A               CPI   :0A
EB1A DA24EB             JC    :EB24    O.K. if number 0-9
EB1D D607               SUI   :07
EB1F FE0A               CPI   :0A
EB21 D8                 RC             Error if #3A-#3F
EB22 FE10               CPI   :10
EB24 3F         ASHCC   CMC            O.K. if 0-9, A-F
EB25 C9                 RET
                *
                *
                *
EB26                    END
```

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

ADACE  EAEB   ADACL  EAE1   ADADC  EAF4   ADALT  EADB
ADARG  EADE   ADART  EB07   ASHCC  EB24   ASHEX  EB15
CALRX  EA0D   CMDTB  EA8E   DISL1  EABE   DISL2  EAC4
DISP   EABE   DISPK  EAB3   ERROR  EA62   ERRST  EA6A
L3E158 EA00   L3E159 EA04   L3E160 EA09   L3E162 EA3C
L3E163 EA42   L3E164 EA54   MSGSO  EA7D   RESET  EA74
```

```
002                             ORG   :EB26
003                     *
004                     *
005                     *
006                     ************
007                     * L - LOOK *
008                     ************
009                     *
010 EB26 0E03           LOOKK   MVI   C,:03     Nr datablocks allowed
011 EB28 CD07EB                 CALL  :EB07     Scan keyb, display char,
012                                             get addresses on stack
013 EB2B 2A5D00                LHLD  :005D     Get addr next instr
014 EB2E EB                     XCHG            in DE
015 EB2F 79                     MOV   A,C       Get nr of inputs done
016 EB30 FE03                   CPI   :03       No addr given?
017 EB32 3E00                   MVI   A,:00
018 EB34 CC56EB                 CZ    :EB56     No addr: Check CR given
019 EB37 C441EB                 CNZ   :EB41     Else: store windows and
020                                             start program
021 EB3A 325000                 STA   :0050     Set Look flag
022 EB3D EB                     XCHG            Addr next instr in HL
023 EB3E C335EF                 JMP   :EF35     Init RST 0
024                     *
025                     * SET LOOK WINDOWS, START LOOK:
026                     *
027                     * Entry: A=0, C=3 minus number of fields read.
028                     * Exit:  Input 2 fields: A,C = 0.
029                     *        Input 3 fields: A,C = FF. DE corrupted.
030                     *        B preserved, HL corrupted.
031                     *
032 EB41 0D             L3E178  DCR   C
033 EB42 0D                     DCR   C
034 EB43 CA62EA                 JZ    :EA62     Error if only 1 addr given
035 EB46 E1                     POP   H         Get returnaddr from stack
036 EB47 E3                     XTHL            haddr window in HL
037 EB48 224800                 SHLD  :0048     Save haddr
038 EB4B E1                     POP   H         Return addr from stack
039 EB4C E3                     XTHL            laddr window in HL
040 EB4D 224A00                 SHLD  :004A     Save laddr
041 EB50 0C                     INR   C
042 EB51 C8                     RZ              Ready if only window given
043
044                     * If startaddress given:
045
046 EB52 3D                     DCR   A         A=FF
047 EB53 E1                     POP   H         Get returnaddr from stack
048 EB54 D1                     POP   D         Get startaddr program in DE
049 EB55 E9                     PCHL            Return
050                     *
051                     ********************************
052                     * GO/LOOK: CHECK FOR CAR.RET *
053                     ********************************
054                     *
055                     * Entry: B: Character to be checked.
056                     * Exit:  AF corrupted, BCDEHL preserved.
057                     *
058 EB56 78             L3E179  MOV   A,B       Get last char typed in
059 EB57 D60D                   SUI   :0D
060 EB59 C262EA                 JNZ   :EA62     Error if not CR
061 EB5C C9                     RET
062                     *
063                     *
```

```
064                     **********************
065                     * RESTART 0 (RST 0) *
066                     **********************
067                     *
068                     * The RST 0 function is used to operate 'LOOK'.
069                     * Via a timer 1 interrupt, RST 0 is called.
070                     * The vectoraddress #EB5D must have been
071                     * initialised by a Z2 or Z3 command.
072                     *
073                     * On entry, HL and PC are on stack (see 0000
074                     * -0007).
075                     *
076 EB5D F5              L3E180  PUSH  PSW
077 EB5E 00                      NOP
078 EB5F 3A4700                 LDA   :0047     Get stored EI/DI
079 EB62 D6FB                   SUI   :FB
080 EB64 CA6CEB                 JZ    :EB6C     Jump if EI stored
081 EB67 3E01                   MVI   A,:01     Else: Set int. mask
082 EB69 32F8FF                 STA   :FFF8     for timer 1 only
083
084                     * Save all registers in RAM area:
085
086 EB6C FB              L3E181  EI
087 EB6D F1                     POP   PSW       Restore PSW
088 EB6E E1                     POP   H         Restore HL
089 EB6F 225900                 SHLD  :0059     Save HL
090 EB72 F5                     PUSH  PSW
091 EB73 C5                     PUSH  B
092 EB74 D5                     PUSH  D
093 EB75 E1                     POP   H
094 EB76 225700                 SHLD  :0057     Save DE
095 EB79 E1                     POP   H
096 EB7A 225500                 SHLD  :0055     Save BC
097 EB7D E1                     POP   H
098 EB7E 225300                 SHLD  :0053     Save PSW
099 EB81 E1                     POP   H
100 EB82 225D00                 SHLD  :005D     Save PC
101
102                     * Check if PC points to 004D-4E:
103
104 EB85 11B3FF                 LXI   D,:FFB3
105 EB88 19                     DAD   D
106 EB89 EB                     XCHG            DE=PC+FFB3 (=PC-004D)
107 EB8A 2A5100                 LHLD  :0051     Get addr where to continue
108 EB8D 7A                     MOV   A,D
109 EB8E B7                     ORA   A
110 EB8F C298EB                 JNZ   :EB98     Jump if out of interrupt
111                                             routine
112 EB92 B3                     ORA   E
113 EB93 FE04                   CPI   :04
114 EB95 DA3EEC                 JC    :EC3E     Jump if PC = 004D-4E
115
116                     * Put returnaddress on stack if current instruction
117                     * is a RST or a CALL instruction:
118
119 EB98 06C7            L3E182  MVI   B,:C7
120 EB9A 7E                     MOV   A,M       Get instr code of next instr
121 EB9B FECD                   CPI   :CD
122 EB9D CAACEB                 JZ    :EBAC     Jump if it is a CALL
123 EBA0 A0                     ANA   B         ) Check if it is a RST
124 EBA1 B8                     CMP   B         )
125 EBA2 CAAEEB                 JZ    :EBAE     Then jump
```

```
126 EBA5 7E                 MOV   A,M        Get instr code
127 EBA6 A0                 ANA   B          ) Check if it is a
128 EBA7 FEC4               CPI   :C4        ) conditional CALL
129 EBA9 C2B0EB             JNZ   :EBB0      Jump if not
130 EBAC 23         L3E183  INX   H
131 EBAD 23                 INX   H
132 EBAE 23         L3E184  INX   H
133 EBAF E3                 XTHL             Addr next instr on stack
134                                          if it was a RST or CALL
135
136                 * Check if current instruction is inside window:
137
138 EBB0 210000      L3E185 LXI   H,:0000
139 EBB3 39                 DAD   SP
140 EBB4 225B00             SHLD  :005B      Save SP
141 EBB7 CDE7ED             CALL  :EDE7      Exchange bytes in reg. save
142                                          area
143 EBBA 2A5100             LHLD  :0051      Get addr current instr
144 EBBD EB                 XCHG             in DE
145 EBBE 2A4A00             LHLD  :004A      Get laddr window
146 EBC1 CD45EC             CALL  :EC45      Compare DE-HL
147 EBC4 FADCEB             JM    :EBDC      Jump if addr outside window
148 EBC7 2A4800             LHLD  :0048      Get haddr window
149 EBCA CD45EC             CALL  :EC45      Compare DE-HL
150 EBCD D2DCEB             JNC   :EBDC      Jump if addr outside window
151
152                 * Print registers contents if address inside window:
153
154 EBD0 CD3AED             CALL  :ED3A      Print car.ret
155 EBD3 115100             LXI   D,:0051    Startaddr reg. save area
156 EBD6 219CEE             LXI   H,:EE9C    Startaddr symbol table
157 EBD9 CD44EE             CALL  :EE44      Print reg. contents
158 EBDC CDC2EE     L3E186  CALL  :EEC2      Scan for Break pressed;
159                                          evt run Break
160 EBDF 2A5D00             LHLD  :005D      Get addr next instr
161
162                 * Disable UT to trace itself:
163                 * Entry from init, RST0.
164                 * HL is address where to continu.
165
166 EBE2 3EEA       L3E187  MVI   A,:EA
167 EBE4 BC                 CMP   H          Check if hibyte = EA
168 EBE5 C2EEEB             JNZ   :EBEE      Jump if not
169 EBE8 3E0D               MVI   A,:0D
170 EBEA BD                 CMP   L          Check if lobyte = 0D
171 EBEB CA62EA             JZ    :EA62      Then go to Error
172
173                 * Check if next opcode is RST or EI/DI:
174
175 EBEE F3       L3E188  DI
176 EBEF 225100             SHLD  :0051      Save addr current instr
177 EBF2 114C00             LXI   D,:004C
178 EBF5 EB                 XCHG
179 EBF6 225D00             SHLD  :005D      Set addr next instr = 004C
180 EBF9 EB                 XCHG
181 EBFA 7E                 MOV   A,M        Get opcode of instr
182 EBFB E6C7               ANI   :C7
183 EBFD FEC7               CPI   :C7
184 EBFF CA33EC             JZ    :EC33      Jump if RST
185 EC02 3A5F00             LDA   :005F      Get int. mask
186 EC05 CD3EEF             CALL  :EF3E      Store it in TIC; set A=0
                            STA   :FFF9      Reset timer 1 (trap
```

```
188                                             immediate)
189 EC0B 7E                 MOV   A,M           Get opcode of instr
190 EC0C E6F7               ANI   :F7           ) Check if EI or DI
191 EC0E D6F3               SUI   :F3           )
192 EC10 7E                 MOV   A,M           Get inst code
193 EC11 EB                 XCHG                HL = #004C
194 EC12 36FB               MVI   M,:FB         Load EI in #004C
195 EC14 0E03               MVI   C,:03         3 instr bytes to be loaded
196                                             into #004D-F
197 EC16 CA2CEC             JZ    :EC2C         Jump if instr is EI/DI
198
199                 * Load next instruction into 004D-4F:
200
201 EC19 1A          L3E189 LDAX  D             Get instr code
202 EC1A 23          L3E190 INX   H
203 EC1B 77                 MOV   M,A           Store it in #004D-F
204 EC1C 13                 INX   D
205 EC1D 0D                 DCR   C
206 EC1E C219EC             JNZ   :EC19         Next byte of instr
207                 *
208                 * Run LOOK: Register contents:
209                 *
210                 *   If 'normal instr': 004C:    EI.
211                 *                      004D-4F: Next instruction.
212                 *
213                 *   If RST *:          004C-4D: RST */ data **.
214                 *                      004E:    RST 0.
215                 *
216                 *   If EI/DI:          004C:    EI.
217                 *                      004D:    NOP.
218                 *                      004E-50: Next instruction.
219                 *                         This may cause problems,
220                 *                         because 0050 is LOOK init
221                 *                         flag.
222                 *
223 EC21 CD30EF             CALL  :EF30         XRA A; LXI H,#0050
224 EC24 BE                 CMP   M             Check look flag
225 EC25 77                 MOV   M,A           Reset flag
226 EC26 CA63EC             JZ    :EC63         If not Look init: restore
227                                             CPU reg; cont on (005D/E)=
228                                             004C
229 EC29 C39CED             JMP   :ED9C         Else: Restore TIC/GICC/
230                                             CPU reg/int.mask  and 'GO'
231                                             to (005D/E)=004C
232
233                 * If instruction is EI or DI:
234
235 EC2C 324700        L3E191 STA   :0047       Store EI/DI instr
236 EC2F AF                 XRA   A
237 EC30 C31AEC             JMP   :EC1A         Load 00 in #004D, next instr
238                                             in 004E-0050
239
240                 * If a RST instruction:
241
242 EC33 EB          L3E192 XCHG                HL=004C
243 EC34 0E02               MVI   C,:02         Only 2-byte instruction
244 EC36 3EC7               MVI   A,:C7
245 EC38 324E00             STA   :004E         Set (004E)=#C7 (=RST0)
246 EC3B C344EF             JMP   :EF44         RST*/data ** in 004C/D
247
248                 * If PC points to 004D-004E:
249
```

```
250 EC3E 19          L3E193  DAD   D          Set HL=0000-0001. This may
251                                           cause re-entry problems due
252                                           to stack manipulation
253 EC3F 225D00              SHLD  :005D      Save addr next instr (pnts
254                                           to RST0)
255 EC42 C3B0EB              JMP   :EBB0
256                  *
257                  *
258                  *
259 EC45                     END
```

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

```
L3E178 EB41   L3E179 EB56   L3E180 EB5D   L3E181 EB6C
L3E182 EB98   L3E183 EBAC   L3E184 EBAE   L3E185 EBB0
L3E186 EBDC   L3E187 EBE2   L3E188 EBEE   L3E189 EC19
L3E190 EC1A   L3E191 EC2C   L3E192 EC33   L3E193 EC3E
LOOKK  EB26
```

```
002                             ORG   :EC45
003                     *
004                     *
005                     *
006                     ************************
007                     * COMPARE DE WITH HL   *
008                     ************************
009                     *
010                     * Exit: AF corrupted, BCDEHL preserved.
011                     *       HL > DE : A=-1, CY=1, Z=0, S=1.
012                     *       HL = DE : A= 0, CY=0, Z=1, S=0.
013                     *       HL < DE : A= 1, CY=0, Z=0, S=0.
014                     *
015 EC45 7B             L3E194  MOV   A,E
016 EC46 95                     SUB   L
017 EC47 7A                     MOV   A,D
018 EC48 9C                     SBB   H
019 EC49 9F                     SBB   A
020 EC4A F8                     RM                  If HL>DE: A=1, S=1
021 EC4B 7B                     MOV   A,E
022 EC4C AD                     XRA   L
023 EC4D D5                     PUSH  D
024 EC4E 5F                     MOV   E,A
025 EC4F 7A                     MOV   A,D
026 EC50 AC                     XRA   H
027 EC51 B3                     ORA   E
028 EC52 D1                     POP   D
029 EC53 37                     STC
030 EC54 C8                     RZ                  If HL=DE: A=0, Z=CY=1
031 EC55 AF                     XRA   A
032 EC56 3C                     INR   A
033 EC57 C9                     RET                 If HL<DE: A=1, CY=0
034                     *
035                     ************************************
036                     * DECREMENT HL, COMPARE HL WITH DE *
037                     ************************************
038                     *
039                     * HL = HL - 1.
040                     *
041                     * Exit: AF corrupted, BCDEHL preserved.
042                     *       HL was 0: CY=1, Z=1.
043                     *       Else:     New HL > DE : CY=0.
044                     *                 New HL = DE :  Z=1.
045                     *                 New HL < DE : CY=1.
046                     *
047 EC58 2B             L3E195  DCX   H
048 EC59 7C                     MOV   A,H
049 EC5A A5                     ANA   L
050 EC5B C601                   ADI   :01
051 EC5D D8                     RC                Ready if old HL was 0
052 EC5E 7D                     MOV   A,L         )
053 EC5F 93                     SUB   E           ) Compare HL - DE
054 EC60 7C                     MOV   A,H         )
055 EC61 9A                     SBB   D           )
056 EC62 C9                     RET
057                     *
058                     ************************
059                     * RESTORE CPU REGISTERS *
060                     ************************
061                     *
062                     * Restores CPU registers AFBCDEHL and PC.
063                     * Continues at PC address.
```

```
064                     * Stackpointer, TICC and GIC are not restored !
065                     *
066 EC63 CDE7ED         L3E196  CALL   :EDE7      Exchange bytes in reg.
067                                               save area
068 EC66 2A5300         L3E197  LHLD   :0053      Get stored PSW
069 EC69 E5                     PUSH   H
070 EC6A F1                     POP    PSW        Restore PSW
071 EC6B 2A5500                 LHLD   :0055
072 EC6E 44                     MOV    B,H
073 EC6F 4D                     MOV    C,L        Restore BC
074 EC70 2A5700                 LHLD   :0057
075 EC73 EB                     XCHG              Restore DE
076 EC74 2A5D00                 LHLD   :005D
077 EC77 E5                     PUSH   H          Addr next instr on stack
078 EC78 2A5900                 LHLD   :0059      Restore HL
079 EC7B C9                     RET               Goto addr in (005D/E)
080                     *
081                     **********************
082                     * CALCULATE DE - HL *
083                     **********************
084                     *
085                     * Exit: HL = DE - HL.
086                     *       BCDE preserved, AF corrupted.
087                     *
088 EC7C 7B             L3E198  MOV    A,E
089 EC7D 95                     SUB    L
090 EC7E 6F                     MOV    L,A        L=E-L
091 EC7F 7A                     MOV    A,D
092 EC80 9C                     SBB    H
093 EC81 67                     MOV    H,A        H=D-H-CY
094 EC82 C9                     RET
095                     *
096                     ************
097                     * M - MOVE *
098                     ************
099                     *
100                     * Moves a block of data (laddr - haddr) given to
101                     * a given destination address (daddr).
102                     *
103                     * Exit: BC: 1st unused destination address.
104                     *       DE: Last source address for direction of
105                     *           movement.
106                     *       HL: 1st unused source address.
107                     *       AF: Corrupted.
108                     *
109 EC83 0E03           MOVEK   MVI    C,:03      Nr of addr allowed
110 EC85 CDDEEA                 CALL   :EADE      Get addr on stack
111 EC88 0D                     DCR    C          3 addr given ?
112 EC89 F262EA                 JP     :EA62      Error if not
113 EC8C C1                     POP    B          daddr in BC
114 EC8D D1                     POP    D          haddr in DE
115 EC8E E1                     POP    H          laddr in HL
116 EC8F E5                     PUSH   H          Save laddr on stack
117 EC90 CD7CEC                 CALL   :EC7C      Calc length of block to
118                                               be moved
119 EC93 DA62EA                 JC     :EA62      Error if wrong inputs
120 EC96 E3                     XTHL              laddr in HL, length on stack
121 EC97 79                     MOV    A,C        )
122 EC98 95                     SUB    L          ) Check if daddr < laddr
123 EC99 78                     MOV    A,B        )
124 EC9A 9C                     SBB    H          )
125 EC9B DAAEEC                 JC     :ECAE      Then jump
```

```
126
127                     * If daddr > laddr:
128
129 EC9E E3                      XTHL            length in HL, laddr on stack
130 EC9F 09                      DAD   B         daddr of highest byte
131 ECA0 44                      MOV   B,H       ) Store it in BC
132 ECA1 4D                      MOV   C,L       )
133 ECA2 E1                      POP   H         laddr in HL
134 ECA3 EB                      XCHG            DE: laddr, HL: haddr
135 ECA4 7E             L3E200   MOV   A,M       Get byte to be moved
136 ECA5 02                      STAX  B         Move it
137 ECA6 0B                      DCX   B         Decr pntr daddr
138 ECA7 CD58EC                  CALL  :EC58     Check if ready
139 ECAA D8                      RC              Then quit
140 ECAB C3A4EC                  JMP   :ECA4     Next byte
141
142                     * If daddr < laddr:
143
144 ECAE E3             L3E201   XTHL            Length in HL, laddr on stack
145 ECAF E1                      POP   H         laddr in HL
146 ECB0 7E             L3E202   MOV   A,M       Get byte
147 ECB1 02                      STAX  B         And move it
148 ECB2 03                      INX   B         Incr pntr daddr
149 ECB3 CD80ED                  CALL  :ED80     INX H; check if ready
150 ECB6 D8                      RC              Then quit
151 ECB7 C3B0EC                  JMP   :ECB0     Next byte
152                     *
153                     *************
154                     * Z - RESET *
155                     *************
156                     *
157                     * Z1: Reset CPU save area 0051-005E. Initialise
158                     *     stackpointer to F900 and save it in 005B/C.
159                     *
160                     * Z2: Sets: current interrupt mask (005F) = #C5,
161                     *           TICC control word (0060) = #FC,
162                     *           GIC  control word (0061) = #1B.
163                     *     Initialises interrupt vector area #0062-#0071.
164                     *     Sets interrupt vector RST 0 to #EB5D.
165                     *
166                     * Z3: Z1 + Z2.
167                     *
168                     * Exit: All registers corrupted.
169                     *
170 ECBA 0E01           ZEROK    MVI   C,:01     Nr of databytes allowed
171 ECBC CDDEEA                  CALL  :EADE     Get hexnr and store it
172                                             on stack
173 ECBF E1                      POP   H         Get hexnr from stack
174 ECC0 7D                      MOV   A,L       into A
175 ECC1 F5                      PUSH  PSW       Save it again
176 ECC2 E602                    ANI   :02
177 ECC4 CAE9EC                  JZ    :ECE9     Jump if Z1 only
178
179                     * If Z2 or Z3:
180
181 ECC7 3EC5                    MVI   A,:C5     Set interrupt mask for
182                                             clock, keyb, ext, timer 1
183 ECC9 325F00                  STA   :005F     Preserve int.mask
184 ECCC 3EF4                    MVI   A,:F4
185 ECCE 00                      NOP
186 ECCF 00                      NOP
187 ECD0 00                      NOP
```

```
188 ECD1 F608                ORI   :08
189 ECD3 216000              LXI   H,:0060
190 ECD6 77                  MOV   M,A        Set TICC contr.word = FC
191 ECD7 23                  INX   H
192 ECD8 361B                MVI   M,:1B      Set GIC contr.word = 1B
193 ECDA CD1ED9              CALL  :D91E      Init int.vector addr
194 ECDD 215DEB              LXI   H,:EB5D
195 ECE0 226200              SHLD  :0062      Set RST0 vector = EB5D
196 ECE3 00                  NOP
197 ECE4 00                  NOP
198 ECE5 00                  NOP
199 ECE6 00                  NOP
200 ECE7 00                  NOP
201 ECE8 00          L3E204  NOP
202 ECE9 F1          ZER01   POP   PSW        Get nr Z instr back
203 ECEA 1F                  RAR              Check for Z2 only
204 ECEB D0                  RNC              Ready if Z2 only
205
206                  * If Z1 or Z3:
207
208 ECEC 0E00                MVI   C,:00      )
209 ECEE 115E00              LXI   D,:005E    ) Init reg. save area:
210 ECF1 215100              LXI   H,:0051    )
211 ECF4 CD54ED              CALL  :ED54      ) Fill (0051-005E) with 00
212 ECF7 2100F9              LXI   H,:F900
213 ECFA F9                  SPHL             Set SP=F900
214 ECFB 225B00              SHLD  :005B      Save SP
215 ECFE C342EA              JMP   :EA42      Return for new inputs
216                  *
217                  ****************
218                  * PRINT SPACE *
219                  ****************
220                  *
221                  * Exit: AFC corrupted, BDEHL preserved.
222                  *
223                  TSP
224 ED01 0E20        LSP     MVI   C,:20
225 ED03 C3B4EE              JMP   :EEB4      Print space
226                  *
227                  ************************************
228                  * SCAN KEYBOARD, PRINT CHARACTER *
229                  ************************************
230                  *
231                  * Gets a keyboard input. Zeroes are ignored.
232                  * The character is printed (if not 'ESC').
233                  *
234                  * Exit: A : Character typed in.
235                  *       Other registers preserved.
236                  *
237 ED06 CDB8EE      CIE     CALL  :EEB8      Scan keyboard
238 ED09 E67F                ANI   :7F        Skip bit 7
239 ED0B CA06ED              JZ    :ED06      Ignore zeroes (await an
240                                           input)
241 ED0E FE12                CPI   :12        'ESC' ?
242 ED10 C5                  PUSH  B
243 ED11 4F                  MOV   C,A        Char in C
244 ED12 C4B4EE              CNZ   :EEB4      Print char if not 'ESC'
245 ED15 79                  MOV   A,C        Char in A
246 ED16 C1                  POP   B
247 ED17 C9                  RET
248                  *
```

```
                        ****************************
                        * PRINT ADDRESS IN ASCII *
                        ****************************
                        *
                        * LADDR: An address in HL is converted to ASCII and
                        *        printed (4 nibbles).
                        * LBYTE: A 1-byte value is printed in ASCII.
                        *
                        * Entry: LADDR: Address in HL.
                        *        LBYTE: Value in A.
                        * Exit:  AFC corrupted, BDEHL preserved.
                        *
ED18 7C         LADDR   MOV   A,H       Hibyte in A.
ED19 CD1DED             CALL  :ED1D     Print both nibbles in ASCII
ED1C 7D                 MOV   A,L       Lobyte in A
ED1D F5         LBYTE   PUSH  PSW       Preserve hex value 2 char
ED1E 07                 RLC             )
ED1F 07                 RLC             ) Move hinibble
ED20 07                 RLC             ) into lonibble
ED21 07                 RLC             )
ED22 CD26ED             CALL  :ED26     Print lonibble
ED25 F1                 POP   PSW       Restore byte
ED26 E60F       L3E210  ANI   :0F       Lonibble only
ED28 CD40ED             CALL  :ED40     Convert it to ASCII
ED2B 4F                 MOV   C,A       And store it in C
ED2C C3B4EE             JMP   :EEB4     Print char in C
                        *
                        ****************
                        * PRINT STRING *
                        ****************
                        *
                        * Entry: HL points to string. End of string is 00.
                        * Exit:  HL points to 00 at end of string.
                        *        AFC corrupted, BDE preserved.
                        *
ED2F 7E         L3E211  MOV   A,M       Get byte from string
ED30 4F                 MOV   C,A       into C
ED31 B7                 ORA   A         Byte = 00 ?
ED32 C8                 RZ              Then ready
ED33 CDB4EE             CALL  :EEB4     Print char in C
ED36 23                 INX   H         Pnts to next char
ED37 C32FED             JMP   :ED2F     Print next char
                        *
                        ************************
                        * PRINT CARRIAGE RETURN *
                        ************************
                        *
                        * Exit: AFC corrupted, BDEHL preserved.
                        *
ED3A 0E0D       LCRLF   MVI   C,:0D
ED3C CDB4EE             CALL  :EEB4     Print 'CR'
ED3F C9                 RET
                        *
                        ******************************
                        * CONVERT HEX NIBBLE TO ASCII *
                        ******************************
                        *
                        * Entry: Hex value in A.
                        * Exit:  ASCII value in A.
                        *        BCDEHL preserved. F corrupted.
                        *
ED40 C630       L3E213  ADI   :30       Convert
```

```
312 ED42 FE3A              CPI   :3A       Number 0-9?
313 ED44 D8                RC             Then ready
314 ED45 C607              ADI   :07       Convert A-F
315 ED47 C9                RET
316                  *
317                  ************
318                  * F - FILL *
319                  ************
320                  *
321                  * Fills a memory area between given boundaries
322                  * with given data.
323                  *
324                  * Exit: CY=1. All registers corrupted.
325                  *
326 ED48 0E03        FILLK MVI   C,:03     Nr of addr/data allowed
327 ED4A CDDEEA             CALL  :EADE     Get addr/data on stack
328 ED4D 0D                DCR   C         3 blocks given?
329 ED4E F262EA            JP    :EA62     Error if not
330 ED51 C1                POP   B         Data in C
331 ED52 D1                POP   D         haddr in DE
332 ED53 E1                POP   H         laddr in HL
333 ED54 71          FILL  MOV   M,C       Data into memory
334 ED55 CD80ED            CALL  :ED80     INX H; check if ready
335 ED58 D8                RC             Then quit
336 ED59 C354ED            JMP   :ED54     Fill next addr
337                  *
338                  ******************
339                  * S - SUBSTITUTE *
340                  ******************
341                  *
342 ED5C 0E01        SUBSK MVI   C,:01     Nr of addr allowed
343 ED5E CDDEEA            CALL  :EADE     Get addr on stack
344 ED61 E1                POP   H         Addr in HL
345 ED62 7E                MOV   A,M       Get contents of addr
346 ED63 CD1DED            CALL  :ED1D     Print it in ASCII
347 ED66 AF                XRA   A
348 ED67 CD6AEE            CALL  :EE6A     Evt. modify contents
349 ED6A D262ED            JNC   :ED62     Next addr
350 ED6D C9                RET
351                  *
352                  ***************
353                  * X - EXAMINE *
354                  ***************
355                  *
356 ED6E 219DEE      EXAMK LXI   H,:EE9D   Startaddr register table
357 ED71 115300            LXI   D,:0053   Startaddr CPU save area
358 ED74 C339EE            JMP   :EE39     Go to display routine
359                  *
360                  ************************
361                  * V - VECTOR EXAMINE *
362                  ************************
363                  *
364 ED77 21A8EE      VECXK LXI   H,:EEA8   Startaddr vector table
365 ED7A 115F00            LXI   D,:005F   Startaddr vector area
366 ED7D C339EE            JMP   :EE39     Go to display routine
367                  *
368                  ************************************
369                  * INCREMENT HL AND COMPARE WITH DE *
370                  ************************************
371                  *
372                  * Exit: HL was FFFF: CY=1, Z=1.
373                  *       Else: New HL < DE : CY=0.
```

```
374                     *               New HL = DE :  Z=1.
375                     *               New HL > DE : CY=1.
376                     *       BCDE preserved, HL=HL+1, AF corrupted.
377                     *
378 ED80 23             INXCK   INX   H
379 ED81 37                     STC                   CY=1
380 ED82 7C                     MOV   A,H
381 ED83 B5                     ORA   L
382 ED84 C8                     RZ                    Abort if new HL is 0000
383 ED85 7B                     MOV   A,E
384 ED86 95                     SUB   L
385 ED87 7A                     MOV   A,D
386 ED88 9C                     SBB   H
387 ED89 C9                     RET
388                     *
389                     **********
390                     * G - GO *
391                     **********
392                     *
393                     * Reads one field if given.
394                     * If no field given: Restores CPU registers, goes
395                     *    to PC address. Returnaddress is #EA42 (into
396                     *    command loop).
397                     * If field given: Saves it as PC, initialises TICC
398                     *    and GIC. Returnaddress is #EA04. Goes to the
399                     *    given address.
400                     * REMARK: The stackpointer is never restored !
401                     *
402 ED8A 0E01           GOK     MVI   C,:01
403 ED8C CD07EB                 CALL  :EB07           Scan keyb; addr on stack
404 ED8F 0D                     DCR   C               No addr given?
405 ED90 F5                     PUSH  PSW
406 ED91 CC56EB                 CZ    :EB56           Then check if 'CR'
407 ED94 F1                     POP   PSW
408 ED95 CA63EC                 JZ    :EC63           No addr: restore CPU reg
409                                                   (but not SP/TICC/GIC !) and
410                                                   go to addr in 005D/E
411 ED98 E1                     POP   H               'GO' addr in HL
412 ED99 225D00                 SHLD  :005D           Save it
413 ED9C 2A6000         L3E221  LHLD  :0060           Get GIC/TICC init values
414 ED9F 7C                     MOV   A,H             GIC init value in A
415 EDA0 CDD5ED                 CALL  :EDD5           Init GIC
416 EDA3 7D                     MOV   A,L             TICC init value in A
417 EDA4 CDB7ED                 CALL  :EDB7           Init TICC
418 EDA7 3A5F00                 LDA   :005F           Get current int mask
419 EDAA 32F8FF                 STA   :FFF8           Set int mask
420 EDAD CDE7ED                 CALL  :EDE7           exch. bytes in 0051-5E
421 EDB0 2104EA                 LXI   H,:EA04         Returnaddr for 'GO'
422 EDB3 E5                     PUSH  H               Save EA04 on stack
423 EDB4 C366EC                 JMP   :EC66           Restore CPU reg and 'GO'
424                     *
425                     * INITIALISE TICC:
426                     *
427                     * Entry: A: Initial value TICC command word (#FC).
428                     * Exit:  AFBC corrupted, DEHL preserved.
429                     *
430 EDB7 47             L3E222  MOV   B,A             Init.value in B
431 EDB8 07                     RLC                   A=F9
432 EDB9 F5                     PUSH  PSW             On stack: A=F9, CY=1
433 EDBA 07                     RLC                   A=F3
434 EDBB 07                     RLC                   A=E7
435 EDBC 07                     RLC                   A=CF
```

```
436 EDBD E607                ANI   :07       A=07
437 EDBF 4F                  MOV   C,A       C=07
438 EDC0 AF                  XRA   A         A=0
439 EDC1 37                  STC             CY=1
440 EDC2 17          L3E223  RAL             ) On exit: A=80,
441 EDC3 0D                  DCR   C         ) C=FF, CY=0
442 EDC4 F2C2ED              JP    :EDC2     )
443 EDC7 4F                  MOV   C,A       C=80
444 EDC8 F1                  POP   PSW       A=F9, CY=1
445 EDC9 79                  MOV   A,C       A=80
446 EDCA 1F                  RAR             A=C0
447 EDCB 32F5FF              STA   :FFF5     Set comm.rate reg for 9600
448                                          baud, 1 stop bit
449 EDCE 78                  MOV   A,B       A=FC
450 EDCF E60F                ANI   :0F       A=0C
451 EDD1 32F4FF              STA   :FFF4     Set cmd reg for IN7, INTA
452                                          enable
453 EDD4 C9                  RET
454                  *
455                  * INITIALISE GIC:
456                  *
457                  * This initialisation cancels the initial setting
458                  * done during 'power-on' by 3EF90. Only used during
459                  * 'GO'.
460                  * There seems to be some bug in the routine. All
461                  * ports are set to input, and then data is written
462                  * into one of this ports (PB - FE01). The function
463                  * of L3E225 is nonsense (?!).
464                  *
465                  * Entry: A: Initial value GIC command word (#1B).
466                  * Exit:  AFBC corrupted, DEHL preserved.
467                  *
468 EDD5 F5          L3E224  PUSH  PSW       Init value on stack, CY=0
469 EDD6 0103FE              LXI   B,:FE03   Addr command word
470 EDD9 F680                ORI   :80       A=9B
471 EDDB 02                  STAX  B         Set all ports to input
472 EDDC 0D                  DCR   C         BC=FE02
473 EDDD F1                  POP   PSW       A=1B, CY=0
474 EDDE 07                  RLC             A=36, CY=0
475 EDDF 9F                  SBB   A         A=0
476 EDE0 0B          L3E225  DCX   B         BC=FE01
477 EDE1 02                  STAX  B         (FE01)=00 (?!)
478 EDE2 0D                  DCR   C         BC=FE00
479 EDE3 F2E0ED              JP    :EDE0     Writes 00 in non-existing
480                                          address FDFE (?!)
481 EDE6 C9                  RET
482                  *
483                  ******************************************
484                  * EXCHANGE BYTES IN REGISTER SAVE AREA *
485                  ******************************************
486                  *
487                  * The hibytes and the lobytes of the addresses
488                  * 0053/0054 thru 0059/5A are exchanged which
489                  * each other.
490                  *
491                  * Exit: AFBCHL corrupted. DE preserved.
492                  *
493 EDE7 215300      L3E238  LXI   H,:0053   Startaddr
494 EDEA 3E04                MVI   A,:04     Nr of addr to be exchanged
495 EDEC 46          L3E226  MOV   B,M       1st byte in B
496 EDED 23                  INX   H         Pnts to next location
497 EDEE 4E                  MOV   C,M       2nd byte in C
```

```
498 EDEF 70                    MOV   M,B       1st byte in 2nd location
499 EDF0 2B                    DCX   H
500 EDF1 71                    MOV   M,C       2nd byte in 1st location
501 EDF2 23                    INX   H
502 EDF3 23                    INX   H         Points to next addr
503 EDF4 3D                    DCR   A         Update counter
504 EDF5 C2ECED                JNZ   :EDEC     Next addr if not ready
505 EDF8 C9                    RET
506                  *
507                  *
508                  *
509 EDF9                       END
```

****
* SYMBOL TABLE *
****

CIE	ED06	EXAMK	ED6E	FILL	ED54	FILLK	ED48
GOK	ED8A	INXCK	ED80	L3E194	EC45	L3E195	EC58
L3E196	EC63	L3E197	EC66	L3E198	EC7C	L3E200	ECA4
L3E201	ECAE	L3E202	ECB0	L3E204	ECE8	L3E210	ED26
L3E211	ED2F	L3E213	ED40	L3E221	ED9C	L3E222	EDB7
L3E223	EDC2	L3E224	EDD5	L3E225	EDE0	L3E226	EDEC
L3E238	EDE7	LADDR	ED18	LBYTE	ED1D	LCRLF	ED3A
LSP	ED01	MOVEK	EC83	SUBSK	ED5C	TSP	ED01
VECXK	ED77	ZERO1	ECE9	ZEROK	ECBA		

```
002                             ORG   :EDF9
003                     *
004                     *
005                     *
006                     ********************************
007                     * PRINT CONTENTS CPU SAVE AREA *
008                     ********************************
009                     *
010                     * Entry: HL: Points to register save area.
011                     *        msb A = 0: 1 byte to be printed.
012                     *        msb A = 1: 2 bytes to be printed.
013                     * Exit:  msb A = 0: AFC corrupted, BDEHL preserved.
014                     *        msb A = 1: AFCDE corrupted, BHL preserved.
015                     *
016 EDF9 B7             L3E227  ORA   A         Test flags
017 EDFA 7E                     MOV   A,M       Get contents save area
018 EDFB F21DED                 JP    :ED1D     1 byte: print it in ASCII
019
020                     * If 2 bytes:
021
022 EDFE 5E                     MOV   E,M       Get contents in E
023 EDFF 23                     INX   H         Next addr
024 EE00 56                     MOV   D,M       Its contents in D
025 EE01 2B                     DCX   H         Restore HL
026 EE02 E5                     PUSH  H         Save it on stack
027 EE03 EB                     XCHG            Contents addr in HL
028 EE04 CD18ED                 CALL  :ED18     Print 2 bytes in ASCII
029 EE07 E1                     POP   H         Restore HL
030 EE08 C9                     RET
031                     *
032                     ****************************************
033                     * V+X: PRINT ROUTINE IF REGISTER GIVEN *
034                     ****************************************
035                     *
036                     * Entry: HL: Startaddress symbol table.
037                     *        DE: Startaddress CPU save area
038                     *        A : Last input character.
039                     * Exit:  All registers corrupted.
040                     *
041 EE09 47             L3E228  MOV   B,A       Last input in B
042 EE0A CD01ED                 CALL  :ED01     Print space
043 EE0D 7E             L3E229  MOV   A,M       Get symbol from table
044 EE0E E67F                   ANI   :7F       Skip bit 7
045 EE10 CA62EA                 JZ    :EA62     Error if symbol=0
046 EE13 B8                     CMP   B         Compare input with symbol
047 EE14 CA22EE                 JZ    :EE22     Jump if identical
048 EE17 7E                     MOV   A,M       Get symbol
049 EE18 07                     RLC             Check for msb set
050 EE19 23                     INX   H         Next symbol
051 EE1A 13                     INX   D         Next memory area
052 EE1B D20DEE                 JNC   :EE0D     Try next symbol
053 EE1E 13                     INX   D         Next mem. area for '2 byte'
054                                             symbols
055 EE1F C30DEE                 JMP   :EE0D     Try again
056
057                     * If symbol found:
058
059 EE22 D5             L3E230  PUSH  D         Save addr in mem.area
060 EE23 7E             L3E231  MOV   A,M       Get symbol
061 EE24 E3                     XTHL            HL: addr mem.area;
062                                             stack: addr symbol
063 EE25 F5                     PUSH  PSW       Save symbol
```

```
064 EE26 CDF9ED             CALL  :EDF9     Print contents mem.area
065 EE29 F1                 POP   PSW       Retrieve symbol
066 EE2A CD6AEE             CALL  :EE6A     Exchange mem.contents,
067                                         go to next one
068 EE2D DA37EE             JC    :EE37     Jump if 'ESC'
069 EE30 E3                 XTHL            HL: addr symbol,
070                                         stack: next mem.area
071 EE31 23                 INX   H         addr next symbol
072 EE32 7E                 MOV   A,M       Get symbol
073 EE33 B7                 ORA   A         Set flags
074 EE34 C223EE             JNZ   :EE23     Not all symbols done
075                 *
076 EE37 D1         L3E232  POP   D         Retrieve next mem.area
077 EE38 C9                 RET
078                 *
079                 ************************
080                 * V+X: DISPLAY ROUTINE *
081                 ************************
082                 *
083                 * Displays registers at succesive memory locations.
084                 *
085                 * Entry: DE: startaddress memory area to be
086                 *            displayed.
087                 *         HL: Startaddress symbol table.
088                 * Exit:  All registers corrupted.
089                 *
090 EE39 CD06ED     L3E233  CALL  :ED06     Scan keyb, print char
091 EE3C FE0D             CPI   :0D       'CR' ?
092 EE3E C209EE             JNZ   :EE09     Jump if also byte given
093
094                 * If only 'V' or 'X':
095
096 EE41 00                 NOP
097 EE42 00                 NOP
098 EE43 00                 NOP
099 EE44 D5         L3E234  PUSH  D         Save startaddr mem area
100 EE45 4E                 MOV   C,M       Get symbol in C
101 EE46 CDB4EE             CALL  :EEB4     Print symbol
102 EE49 0E3D               MVI   C,:3D
103 EE4B CDB4EE             CALL  :EEB4     Print '='
104 EE4E 7E                 MOV   A,M       Get symbol in A
105 EE4F B7                 ORA   A         Check flags
106 EE50 E3                 XTHL            HL: startaddr mem.area
107                                         stack: startaddr symboltable
108 EE51 F5                 PUSH  PSW       Save symbol + flags
109 EE52 CDF9ED             CALL  :EDF9     Print contents mem.area
110 EE55 F1                 POP   PSW       Retrieve symbol and flags
111 EE56 23                 INX   H
112 EE57 F25BEE             JP    :EE5B     Jump if '1 byte' symbol
113
114                 * If 2-byte symbol:
115
116 EE5A 23                 INX   H
117 EE5B 00         L3E235  NOP
118 EE5C 00                 NOP
119 EE5D 00                 NOP
120 EE5E E3                 XTHL            HL: addr symbol
121                                         stack: addr next mem.area
122 EE5F 23                 INX   H         Next symbol
123 EE60 D1                 POP   D         Get addr next mem.area
124 EE61 7E                 MOV   A,M       Get symbol
125 EE62 B7                 ORA   A
```

```
126 EE63 C8              RZ              Quit if ready
127 EE64 CD01ED          CALL  :ED01     Print space
128 EE67 C344EE          JMP   :EE44     Next one
129                      *
130                      ******************************
131                      * V+X: EVT. MODIFY CONTENTS *
132                      ******************************
133                      *
134                      * Entry: HL: Memory address.
135                      *         A : Symbol.
136                      *
137 EE6A B7              L3E326  ORA   A         Set flags on symbol
138 EE6B F5                      PUSH  PSW       and save it
139 EE6C 0E2D                    MVI   C,:2D
140 EE6E CDB4EE                  CALL  :EEB4     Print '-'
141 EE71 E5                      PUSH  H         Save addr mem.area
142 EE72 0E01                    MVI   C,:01     Nr of datablocks allowed
143 EE74 CD07EB                  CALL  :EB07     Get input on stack; last
144                                              byte typed in in B
145 EE77 0D                      DCR   C
146 EE78 CA87EE                  JZ    :EE87     Jump if incorrect input
147 EE7B D1                      POP   D         Data typed in in DE
148 EE7C E1                      POP   H         Get addr mem.area
149 EE7D F1                      POP   PSW       Get symbol and flags
150 EE7E 73                      MOV   M,E       Change memory contents
151 EE7F F28DEE                  JP    :EE8D     Jump if '1 byte' symbol
152
153                      * If 2-byte symbol:
154
155 EE82 23                      INX   H
156 EE83 72                      MOV   M,D       Change 2nd byte
157 EE84 C38DEE                  JMP   :EE8D
158                      *
159 EE87 E1              L3E327  POP   H         Retrieve addr mem.area
160 EE88 F1                      POP   PSW       Retrieve symbol + flags
161 EE89 F28DEE                  JP    :EE8D     If '1 byte' symbol
162 EE8C 23                      INX   H         Add. INX H if 2-byte symbol
163 EE8D 23              L3E328  INX   H         Next mem.area
164 EE8E 78                      MOV   A,B       Get last input
165 EE8F FE0D                    CPI   :0D
166 EE91 C8                      RZ              Ready if 'CR'
167 EE92 FE20                    CPI   :20
168 EE94 C8                      RZ              Ready if 'SP'
169 EE95 FE12                    CPI   :12
170 EE97 37                      STC             Abort with CY=1 if 'ESC'
171 EE98 C8                      RZ
172 EE99 C362EA                  JMP   :EA62     Else: wrong input: Error
173                      *
174                      ******************************
175                      * SYMBOL TABLE EXAMINE (X) *
176                      ******************************
177                      *
178                      * The msb is '1' for symbols of two-byte
179                      * registers.
180                      *
181 EE9C C9              L3E405  DATA  :C9       I (addr current instr)
182 EE9D 41                      DATA  :41       A
183 EE9E 46                      DATA  :46       F (flags)
184 EE9F 42                      DATA  :42       B
185 EEA0 43                      DATA  :43       C
186 EEA1 44                      DATA  :44       D
187 EEA2 45                      DATA  :45       E
```

```
188 EEA3 48                 DATA  :48       H
189 EEA4 4C                 DATA  :4C       L
190 EEA5 D3                 DATA  :D3       S (stackpointer)
191 EEA6 D0                 DATA  :D0       P (program counter)
192 EEA7 00                 DATA  :00
193                 *
194                 ********************************
195                 * SYMBOL TABLE VECTOR EXAMINE *
196                 ********************************
197                 *
198                 * The msb is '1' for symbols of two-byte
199                 * registers.
200                 *
201 EEA8 4D          L3E406  DATA  :4D       M (TICC int mask)
202 EEA9 54                  DATA  :54       T (TICC cmd + comm.reg)
203 EEAA 47                  DATA  :47       G (GIC cmd word)
204 EEAB B0                  DATA  :B0       0  )
205 EEAC B1                  DATA  :B1       1  )
206 EEAD B2                  DATA  :B2       2  )
207 EEAE B3                  DATA  :B3       3  ) Interrupt vectors
208 EEAF B4                  DATA  :B4       4  ) addresses
209 EEB0 B5                  DATA  :B5       5  )
210 EEB1 B6                  DATA  :B6       6  )
211 EEB2 B7                  DATA  :B7       7  )
212 EEB3 00                  DATA  :00
213                 *
214                 ********************
215                 * PRINT CHARACTER *
216                 ********************
217                 *
218                 * Entry: C: Character to be printed.
219                 * Exit:  AF corrupted, BCDEHL preserved.
220                 *
221 EEB4 79          C0      MOV   A,C       Char in A
222 EEB5 C360DD              JMP   :DD60     Print char
223                 *
224                 ********************************
225                 * SCAN KEYBOARD; IGNORE BREAK *
226                 ********************************
227                 *
228                 * Exit: Character in A.
229                 *       BCDEHL preserved.
230                 *
231 EEB8 CD83EF      C1      CALL  :EF83     Scan keyboard
232 EEBB DAB8EE              JC    :EEB8     Ignore break
233 EEBE CAB8EE              JZ    :EEB8     Wait for input
234 EEC1 C9                  RET
235                 *
236                 ************************************
237                 * SCAN KEYBOARD FOR BREAK PRESSED *
238                 ************************************
239                 *
240                 * Exit: CY=0: No break.
241                 *       CY=1: Break pressed.
242                 *       A corrupted, BCDEHL preserved.
243                 *
244 EEC2 CDA5D6      BREAK   CALL  :D6A5     Scan keyb for new inputs
245 EEC5 D0                  RNC             abort if Break not pressed
246 EEC6 C36AEA              JMP   :EA6A     Restore SP, wait for
247                                           new inputs
248                 *
249                 *
```

```
                        **********************
                        * CASSETTE ROUTINES *
                        **********************
                        *
EEC9 C3C502     L3E335  JMP   :02C5     WOPEN
                        *
EECC FF                 DATA  :FF
EECD FF                 DATA  :FF
EECE FF                 DATA  :FF
                        *
EECF C3C802     L3E336  JMP   :02C8     WBLK
                        *
EED2 FF                 DATA  :FF
EED3 FF                 DATA  :FF
EED4 FF                 DATA  :FF
                        *
EED5 C3CB02     L3E337  JMP   :02CB     WCLOSE
                        *
EED8 C3CE02     L3E338  JMP   :02CE     ROPEN
                        *
EEDB FF                 DATA  :FF
EEDC FF                 DATA  :FF
EEDD FF                 DATA  :FF
                        *
EEDE C3D102     L3E339  JMP   :02D1     RBLK
                        *
EEE1 C3D402     L3E340  JMP   :02D4     RCLOSE
                        *
                        *************
                        * W - WRITE *
                        *************
                        *
                        * Requires 2 address fields + evt. name.
                        * Filetype is '1'. Writes startaddress of datablock
                        * + data + trailer on tape.
                        *
EEE4 0E02       L3E341  MVI   C,:02     Nr of addr allowed
EEE6 CDDEEA             CALL  :EADE     Scan keyb, addr on stack
EEE9 0D                 DCR   C         2 addr given ?
EEEA F262EA             JP    :EA62     Error if not
EEED CD48EF             CALL  :EF48     Evt. name in input buffer
EEF0 3E31               MVI   A,:31     File type byte
EEF2 00                 NOP
EEF3 00                 NOP
EEF4 00                 NOP
EEF5 00                 NOP
EEF6 00                 NOP
EEF7 00                 NOP
EEF8 CDC9EE             CALL  :EEC9     Write file header on tape
EEFB D1                 POP   D         Get haddr from stack
EEFC 13                 INX   D         Incr it
EEFD E1                 POP   H         Get laddr from stack
EEFE CD63EF             CALL  :EF63     Write startaddr on tape
EF01 7B                 MOV   A,E       )
EF02 95                 SUB   L         )
EF03 5F                 MOV   E,A       ) Calc length of data
EF04 7A                 MOV   A,D       ) block, result in DE
EF05 9C                 SBB   H         )
EF06 57                 MOV   D,A       )
EF07 00                 NOP
EF08 CDCFEE             CALL  :EECF     Write datablock on tape
EF0B CDD5EE             CALL  :EED5     Write file trailer
```

```
312 EF0E C9                 RET
313                 *
314                 ************
315                 * R - READ *
316                 ************
317                 *
318                 * One address is allowed. An evt. name is stored
319                 * in the EBUF. Reads header, startaddress and data
320                 * from tape. Errorcheck only on data.
321                 *
322                 * Exit: HL: 1st address above file loaded.
323                 *       BC: Evt. offset.
324                 *       AFDE corrupted.
325                 *
326 EF0F 0E01       RHEXK   MVI   C,:01     Nr of addr allowed
327 EF11 CDDEEA             CALL  :EADE     Scan keyb; addr on stack
328 EF14 CD48EF             CALL  :EF48     Evt name in input buffer
329 EF17 0631             MVI   B,:31     File type byte
330 EF19 00              NOP
331 EF1A 00              NOP
332 EF1B 00              NOP
333 EF1C 0E00            MVI   C,:00
334 EF1E CDD8EE          CALL  :EED8     Read file header
335 EF21 1100F9          LXI   D,:F900   Max addr to write data int
336 EF24 C1              POP   B         Get evt offset from stack
337 EF25 CD74EF          CALL  :EF74     Read startaddr from tape
338 EF28 09              DAD   B         Add offset
339 EF29 CD8AEF          CALL  :EF8A     Read data block + trailer
340 EF2C D262EA          JNC   :EA62     Print 'error' if reading
341                                      error
342 EF2F C9              RET
343                 *
344                 ****************************************
345                 * RST 0: POINTER TO 'LOOK'-FLAG IN HL *
346                 ****************************************
347                 *
348                 * Part of 3EC21.
349                 *
350                 * Exit: A=0, BCDE preserved.
351                 *
352 EF30 AF          L3E343  XRA   A
353 EF31 215000           LXI   H,:0050
354 EF34 C9              RET
355                 *
356                 ********************
357                 * INITIALISE RST 0 *
358                 ********************
359                 *
360                 * Entry: On stack: Returnaddress #EA42.
361                 *
362 EF35 3EFB       L3E344  MVI   A,:FB     Instr code for EI in A
363 EF37 324700           STA   :0047     Save it
364 EF3A D1              POP   D         Get EA42 in DE
365 EF3B C3E2EB          JMP   :EBE2     Into RST 0
366                 *
367                 ************************
368                 * SET INTERRUPT MASK *
369                 ************************
370                 *
371                 * Part of RST0 (3EC05).
372                 *
373                 * Entry: A: Value for TICC interrupt mask.
```

```
374                     * Exit:  A=0, F corrupted, BCDEHL preserved.
375                     *
376 EF3E 32F8FF         L3E345  STA   :FFF8     Set TICC int mask
377 EF41 AF                     XRA   A         A=0
378 EF42 C9                     RET
379                     *
380 EF43 FF                     DATA  :FF
381                     *
382                     ****************************
383                     * part of RST 0 (3EC3B) *
384                     ****************************
385                     *
386 EF44 2B             L3E346  DCX   H
387 EF45 C319EC                 JMP   :EC19     Into RST 0
388                     *
389                     ********************************
390                     * W+R: NAME IN INPUT BUFFER *
391                     ********************************
392                     *
393                     * Names >126 characters destroy BASIC pointers.
394                     *
395                     * Entry: Last character typed in, in B.
396                     * Exit:  BCD preserved. HL points to EBUF.
397                     *        A= 0: No file name given.
398                     *        A<>0: File name given.
399                     *
400 EF48 213E01           L3E347  LXI   H,:013E   Startaddr EBUF
401 EF4B 1EFF                     MVI   E,:FF
402 EF4D 78                       MOV   A,B       Last char in A
403 EF4E D60D                     SUI   :0D       'CR' ?
404 EF50 77                       MOV   M,A       (13E) is 0 if CR
405 EF51 C8                       RZ              Quit if no name given
406
407                     * If name given:
408
409 EF52 E5                       PUSH  H         Save startaddr EBUF
410 EF53 2C             L3E356  INR   L         Points to next loc
411 EF54 1C                     INR   E         Calc length
412 EF55 CD06ED                 CALL  :ED06     Scan keyb, print char
413 EF58 77                     MOV   M,A       Char into EBUF
414 EF59 FE0D                   CPI   :0D
415 EF5B C253EF                 JNZ   :EF53     Next char if not 'CR'
416 EF5E E1                     POP   H         Retrieve startaddr EBUF
417 EF5F 73                     MOV   M,E       Store length name in 013E
418 EF60 C9                     RET
419                     *
420                     ***************
421                     * (Not used) *
422                     ***************
423                     *
424 EF61 E1             L3E348  POP   H
425 EF62 C9                     RET
426                     *
427                     ********************************
428                     * WRITE STARTADDRESS ON TAPE *
429                     ********************************
430                     *
431                     * Entry: HL: Startaddress.
432                     * Exit:  AF corrupted, BCDEHL preserved.
433                     *
434 EF63 D5             L3E349  PUSH  D
435 EF64 E5                     PUSH  H
```

```
436 EF65 223E01             SHLD  :013E     Startaddr in EBUF
437 EF68 213E01             LXI   H,:013E   Startaddr to write from
438 EF6B 110200             LXI   D,:0002   Length
439 EF6E CDCFEE             CALL  :EECF     Write addr on tape
440 EF71 E1                 POP   H
441 EF72 D1                 POP   D
442 EF73 C9                 RET
443                 *
444                 ********************************
445                 * READ STARTADDRESS FROM TAPE *
446                 ********************************
447                 *
448                 * Exit: HL: Startaddress.
449                 *       CY=1: No reading error.
450                 *       CY=0: Reading error, errorcode in A.
451                 *       BCDE preserved.
452                 *
453 EF74 D5        L3E350  PUSH  D
454 EF75 213E01             LXI   H,:013E   Addr EBUF
455 EF78 114101             LXI   D,:0141   Addr after addr in EBUF
456 EF7B CDDEEE             CALL  :EEDE     Read block from tape
457 EF7E D1                 POP   D
458 EF7F 2A3E01             LHLD  :013E     Startaddr in HL
459 EF82 C9                 RET
460                 *
461                 ******************
462                 * SCAN KEYBOARD *
463                 ******************
464                 *
465                 * Part of 3EEB8. Scans keyboard. Returns
466                 * any key received.
467                 *
468                 * Exit: A: Key received.
469                 *       BCDEHL preserved.
470                 *       CY=1: Break pressed.
471                 *
472 EF83 AF        L3E351  XRA   A
473 EF84 32B902             STA   :02B9     Enable complete keyb scan
474 EF87 C3BED6             JMP   :D6BE     Scan keyboard
475                 *
476                 ************************
477                 * READ DATA FROM TAPE *
478                 ************************
479                 *
480                 * Part of READ (3EF29).
481                 *
482 EF8A CDDEEE    L3E352  CALL  :EEDE     Read block from tape
483 EF8D C3E1EE             JMP   :EEE1     Stop reading
484                 *
485                 ********************************
486                 * DCE INITIALISATION ROUTINE *
487                 ********************************
488                 *
489                 * Part of RESET (C719). Bootstrap for disc drive.
490                 * Sets GIC in initialisation status. Checks if any
491                 * input is received from the DCE-bus and performs
492                 * the received instructions.
493                 *
494                 * Exit: A=#EE if no DCE-inputs available.
495                 *
496 EF90 3E98      L3E353  MVI   A,:98
497 EF92 3203FE             STA   :FE03     PA+PCH in, PB+PCL out
```

```
498 EF95 3E07              MVI   A,:07
499 EF97 3203FE            STA   :FE03     PC3=1 —>
500 EF9A 3E01              MVI   A,:01
501 EF9C 3201FE            STA   :FE01     Output PB: 01
502 EF9F 3E01              MVI   A,:01
503 EFA1 3203FE            STA   :FE03     PC0=1
504 EFA4 010010            LXI   B,:1000
505 EFA7 3A02FE    L3E355  LDA   :FE02     Get input from PCH
506 EFAA E620              ANI   :20       Bit 5 only
507 EFAC C2B8EF            JNZ   :EFB8     Jump if inputs received
508
509                * If no inputs:
510
511 EFAF 0B                DCX   B         )
512 EFB0 78                MOV   A,B       ) Wait loop until C=#10
513 EFB1 B1                ORA   C         )
514 EFB2 C2A7EF            JNZ   :EFA7     )
515 EFB5 3EEE              MVI   A,:EE     A=EE if no inputs
516 EFB7 C9                RET
517                *
518                * DCE BOOTSTRAP INPUT ROUTINE:
519                *
520                * Loads MLP inputs from the DCE-bus into the
521                * stackbottom and goes to it.
522                *
523 EFB8 1100F8    L3E354  LXI   D,:F800   Addr stackbottom
524 EFBB 3E05      L3E357  MVI   A,:05
525 EFBD 3203FE            STA   :FE03     PC2=1
526 EFC0 3A02FE    L3E358  LDA   :FE02     Get input from PC
527 EFC3 E680              ANI   :80       Bit 7 only
528 EFC5 CAC0EF            JZ    :EFC0     Wait for change to high
529 EFC8 3A00FE            LDA   :FE00     Get input from PA
530 EFCB 12                STAX  D         Save input in stack area
531 EFCC 13                INX   D         Point to next loc
532 EFCD 3E04              MVI   A,:04
533 EFCF 3203FE            STA   :FE03     PC2=0
534 EFD2 3A02FE    L3E359  LDA   :FE02     Get input from PC
535 EFD5 E680              ANI   :80       Bit 7 only
536 EFD7 C2D2EF            JNZ   :EFD2     Wait for change to low
537 EFDA 3A02FE            LDA   :FE02     Get input from PC
538 EFDD E620              ANI   :20       Bit 5 only
539 EFDF C2BBEF            JNZ   :EFBB     Again if high
540 EFE2 3E06              MVI   A,:06
541 EFE4 323EFE            STA   :FE3E     (FE3E hardwarewise read as
542                                        FE02). PC=06
543 EFE7 3A02FE    L3E360  LDA   :FE02     Get input from PC
544 EFEA E620              ANI   :20       Bit 5 only
545 EFEC CAE7EF            JZ    :EFE7     Wait for change to high
546 EFEF C300F8            JMP   :F800     Go to stackbottom
547                *
548 EFF2 FF                DATA  :FF
549 EFF3 FF                DATA  :FF
550                *
551                **********************
552                * SCAN 'DINC' INPUT *
553                **********************
554                *
555                * Part of 3E935. Default 'DINC' is RS232 input.
556                *
557 EFF4 CDB4DD    L3E361  CALL  :DDB4     Get input from DINC
558 EFF7 CA35E9             JZ    :E935     Scan keyb if no DINC input
```

```
560                    * If inputs from DINC:
561
562 EFFA 3E01                  MVI   A,:01
563 EFFC 329602                STA   :0296      Set INSW for DINC input
564 EFFF C9                    RET
565                    *
566                    *
567                    *
568 F000                       END
```

****
* SYMBOL TABLE *
****

BREAK	EEC2	CI	EEB8	CO	EEB4	L3E227	EDF9
L3E228	EE09	L3E229	EE0D	L3E230	EE22	L3E231	EE23
L3E232	EE37	L3E233	EE39	L3E234	EE44	L3E235	EE5B
L3E326	EE6A	L3E327	EE87	L3E328	EE8D	L3E335	EEC9
L3E336	EECF	L3E337	EED5	L3E338	EED8	L3E339	EEDE
L3E340	EEE1	L3E341	EEE4	L3E343	EF30	L3E344	EF35
L3E345	EF3E	L3E346	EF44	L3E347	EF48	L3E348	EF61
L3E349	EF63	L3E350	EF74	L3E351	EF83	L3E352	EF8A
L3E353	EF90	L3E354	EFB8	L3E355	EFA7	L3E356	EF53
L3E357	EFBB	L3E358	EFC0	L3E359	EFD2	L3E360	EFE7
L3E361	EFF4	L3E405	EE9C	L3E406	EEA8	RHEXK	EF0F