```
002                             ORG   :C000
003                     *
004                     *
005                     *
006                     *  ================
007                     *** MATH. UTILITIES ***
008                     *  ================
009                     *
010                     *
011                     ***************
012                     * ENTRYPOINTS *
013                     ***************
014                     *
015 C000 C319C7         BASE    JMP   :C719     Reset (entry on hardware
016                                               reset)
017 C003 C335C0         XINIT   JMP   :C035     Math. package initialisation
018 C006 C3F3C0         XFINM   JMP   :C0F3     Incr. FPT number in memory
019 C009 C3FBC1         XFDCM   JMP   :C1FB     Decr. FPT number in memory
020                                               (not used)
021 C00C C379C0         XFCOMP  JMP   :C079     FPT Compare
022 C00F C3BBC0         XIINM   JMP   :C0BB     Incr. INT number in memory
023 C012 C3D5C0         XIDCM   JMP   :C0D5     Decr. INT number in memory
024                                               (not used)
025 C015 C3ACC0         XICOMP  JMP   :C0AC     INT Compare
026 C018 C31EC2         XPUSH   JMP   :C21E     Save MACC on stack
027 C01B C334C2         XPOP    JMP   :C234     Retrieve MACC from stack
028 C01E C349C2         XFCB    JMP   :C249     Input FPT number to MACC
029 C021 C361C3         XFBC    JMP   :C361     Conv. FPT number for output
030 C024 C373C5         XICB    JMP   :C573     Input INT number to MACC
031 C027 C3B2C5         XIBC    JMP   :C5B2     Conv. INT number for output
032 C02A C314C6         XHCB    JMP   :C614     Input Hex number to MACC
033 C02D C353C6         XHBC    JMP   :C653     Conv. MACC to Hex for output
034 C030 C386C4         XPRTY   JMP   :C486     Pretties up FPT/INT number
035 C033 E300           DECBUF  DBL   :00E3     Location output buffer
036                     *
037                     ***********************************
038                     * MATH. PACKAGE INITIALISATION *
039                     ***********************************
040                     *
041                     * Entry: HL: Address input encoding routine (DDE0).
042                     *         DE: Base address error routines (C7F2).
043                     * Exit:  AFDEHL corrupted, BC preserved.
044                     *
045 C035 22D200         MINIT   SHLD  :00D2     Init. (00D2/3)=DDE0
046 C038 EB                    XCHG
047 C039 22D000                 SHLD  :00D0     Init. (00D0/1)=C7F2
048 C03C 3A02FB                 LDA   :FB02     Get math.chip status
049 C03F B7                     ORA   A         Check if math.chip present
050 C040 3E00                   MVI   A,:00     Flag = 0 if not
051 C042 FA47C0                 JM    :C047
052 C045 3E7B                   MVI   A,:7B     Flag = #7B if present
053 C047 32D400         LC18    STA   :00D4     Set math.chip flag
054 C04A C9                     RET
055                     *
056                     ***************************
057                     * OVERFLOW ERROR ROUTINE *
058                     ***************************
059                     *
060                     * (From LC20 common part for various entries).
061                     *
062                     * Jump to (00D0/1) = Address 'overflow error'
063                     * routine (C7F2).
```

```
064                     *
065                     * Entry: If start at LC20: offset in HL.
066                     * Exit:  AFBCDEHL preserved.
067                     *        On stack original returnaddress.
068                     *
069 C04B E5             FPEOV   PUSH  H
070 C04C 210000                 LXI   H,:0000   Init offset = 0
071                     *
072 C04F F5             LC20    PUSH  PSW
073 C050 D5                     PUSH  D
074 C051 EB                     XCHG            Offset in DE
075 C052 2AD000                 LHLD  :00D0     Get addr pointer
076 C055 19                     DAD   D         Add offset
077 C056 7E                     MOV   A,M
078 C057 23                     INX   H
079 C058 66                     MOV   H,M
080 C059 6F                     MOV   L,A       Get addr routine in HL
081 C05A D1                     POP   D
082 C05B F1                     POP   PSW
083 C05C E3                     XTHL            New addr on stack
084 C05D C9                     RET             Continu with new address
085                     *
086                     ********************
087                     * ARGUMENT ERROR *
088                     ********************
089                     *
090                     * Jump to (00D0/1)+2 = Address 'number out of range'
091                     * routine (C7F4).
092                     *
093                     * Entry/exit: See FPEOV.
094                     *
095 C05E E5             FPEAE   PUSH  H
096 C05F 210200                 LXI   H,:0002   Init. offset
097 C062 C34FC0                 JMP   :C04F     Calc. new addr, go to it
098                     *
099                     ********************
100                     * UNDERFLOW ERROR *
101                     ********************
102                     *
103                     * Jump to (00D0/1)+4 = Return (C7F6).
104                     * Underflow gives 0 as result of operation.
105                     *
106                     * Entry/exit: See FPEOV.
107                     *
108 C065 E5             FPEUN   PUSH  H
109 C066 215EC4                 LXI   H,:C45E   Addr. FPT(0)
110 C069 C30CD2                 JMP   :D20C     Copy '0' into MACC
111                     *
112                     ************************
113                     * DIVIDE BY ZERO ERROR *
114                     ************************
115                     *
116                     * Jump to (00D0/1)+6 = Address 'divide by zero'
117                     * routine (C7F8).
118                     *
119                     * Entry/exit: See FPEOV.
120                     *
121 C06C E5             FPEDO   PUSH  H
122 C06D 210600                 LXI   H,:0006   Init. offset
123 C070 C34FC0                 JMP   :C04F     Calc. new addr, go to it
124                     *
125                     *
```

```
126                     ****************************
127                     * GET CHARACTER FROM LINE *
128                     ****************************
129                     *
130                     * Entry: None.
131                     * Exit : All registers preserved.
132                     *        Address to continue on stack.
133                     *
134 C073 E5             LC22    PUSH  H
135 C074 2AD200                 LHLD  :00D2     Get addr 'Get char' routine
136 C077 E3                     XTHL            on stack; restore HL
137 C078 C9                     RET             Goto (00D0/1)+2
138                     *
139                     ****************************
140                     * FLOATING POINT COMPARE *
141                     ****************************
142                     *
143                     * Compares normalised FPT numbers in MACC
144                     * and in M.
145                     *
146                     * Exit: ABCDEHL preserved.
147                     *       Flags: CY=1,S=0,Z=1: both nrs. 0
148                     *              CY=0,S=0,Z=1: both nrs. identical
149                     *              CY=0,S=0,Z=0: MACC > M
150                     *              CY=0,S=1,Z=0: MACC < M
151                     *
152 C079 C5             FCOMP   PUSH  B
153 C07A F5                     PUSH  PSW
154 C07B D5                     PUSH  D
155 C07C E5                     PUSH  H
156 C07D E7                     RST   4         Copy MACC to reg A,B,C,D
157 C07E 15                     DATA  :15
158 C07F 5F                     MOV   E,A       Exp.byte in E
159 C080 AE                     XRA   M         XOR both exp.bytes
160 C081 FAB7C0                 JM    :C0B7     Jump if different signs
161
162                     * If equal signs:
163
164 C084 C3E8D1                 JMP   :D1E8     Goto D1E8, return to C087
165                     *
166 C087 17             LC23    RAL
167 C088 C2A3C0                 JNZ   :C0A3
168 C08B 7B             LC24    MOV   A,E
169 C08C 96                     SUB   M         Comp. exp. bytes
170 C08D C2A2C0                 JNZ   :C0A2     Jump if not equal
171 C090 23                     INX   H
172 C091 78                     MOV   A,B
173 C092 96                     SUB   M         Comp. 1st bytes mantissa's
174 C093 C2A2C0                 JNZ   :C0A2     Jump if not equal
175 C096 23                     INX   H
176 C097 79                     MOV   A,C
177 C098 96                     SUB   M         Comp. 2nd bytes mantissa's
178 C099 C2A2C0                 JNZ   :C0A2     Jump if not equal
179 C09C 23                     INX   H
180 C09D 7A                     MOV   A,D
181 C09E 96                     SUB   M         Comp. 3rd bytes mantissa's
182 C09F CAA6C0         LC25    JZ    :C0A6     Jump if not equal
183 C0A2 1F             LC26    RAR             ) Set flags for output
184 C0A3 AB             LC27    XRA   E         )
185 C0A4 F601           LC28    ORI   :01       Clear CY-flag
186 C0A6 E1             LC29    POP   H
187 C0A7 D1                     POP   D
```

```
                    '       POP   B
C0A8 C1             '       MOV   A,B       Restore A
C0A9 78                     POP   B
C0AA C1                     RET
C0AB C9             *
                    ********************
                    * INTEGER COMPARE *
                    ********************
                    *
                    * Compares INT numbers in MACC and M.
                    * REMARK: Routine is incorrect when both
                    *         numbers are negative ! Then result
                    *         is if MACC > M due to LC26/LC27.
                    *
                    * Exit:  ABCDEHL preserved. CY=0
                    *        Flags: S=0, Z=1: Both numbers equal
                    *               S=0, Z=0: MACC > M
                    *               S=1, Z=0: MACC < M
                    *
C0AC C5             ICOMP   PUSH  B
C0AD F5                     PUSH  PSW
C0AE D5                     PUSH  D
C0AF E5                     PUSH  H
C0B0 E7                     RST   4         Copy MACC to reg. A,B,C,D
C0B1 15                     DATA  :15
C0B2 5F                     MOV   E,A       Sign byte in E
C0B3 AE                     XRA   M         XOR both sign bytes
C0B4 F28BC0                 JP    :C08B     If both nrs have same sign:
                                              compare

                    * If different signs:

C0B7 AE             LC241   XRA   M         Find out which one is neg:
                                              S=1: MEM pos; MACC neg
                                              S=0: MEM neg; MACC pos
C0B8 C3A4C0                 JMP   :C0A4     Abort
                    *
                    ****************************************
                    * INCREMENT INTEGER NUMBER IN MEMORY *
                    ****************************************
                    *
                    * Entry: HL points to 1st byte of INT number.
                    * Exit:  All registers preserved.
                    *
C0BB F5             IINM    PUSH  PSW
C0BC E5                     PUSH  H
C0BD 23                     INX   H
C0BE 23                     INX   H
C0BF 23                     INX   H         HL pnts to last byte
C0C0 3E03                   MVI   A,:03     Nr of bytes for INT nr
C0C2 34             LC30    INR   M         Incr. INT nr.
C0C3 C2D2C0                 JNZ   :C0D2     Ready if no overflow
C0C6 2B                     DCX   H         Goto next byte
C0C7 3D                     DCR   A         1st byte reached?
C0C8 C2C2C0                 JNZ   :C0C2     Incr. next byte
C0CB 34                     INR   M         Incr. 1st byte
C0CC 7E                     MOV   A,M       Get it
C0CD FE80                   CPI   :80       msb=1?
C0CF CC4BC0                 CZ    :C04B     Then overflow error
C0D2 E1             LC31    POP   H         Normal return
C0D3 F1                     POP   PSW
C0D4 C9                     RET
```

```
                        *
                        ********************************************************
                        * DECREMENT INTEGER NUMBER IN MEMORY - (not used) *
                        ********************************************************
                        *
                        * Entry: HL points to 1st byte of INT number.
                        * Exit:  All registers preserved.
                        *
C0D5 F5         IDCM    PUSH  PSW
C0D6 C5                 PUSH  B
C0D7 E5                 PUSH  H
C0D8 23                 INX   H
C0D9 23                 INX   H
C0DA 23                 INX   H             HL pnts to last byte
C0DB 0603               MVI   B,:03         Nr. of bytes of INT nr.
C0DD 35         LC32    DCR   M             Decr. INT nr
C0DE 7E                 MOV   A,M
C0DF 3C                 INR   A             Check for overflow
C0E0 C2EFC0             JNZ   :C0EF         Ready if no overflow
C0E3 2B                 DCX   H             Goto next byte
C0E4 05                 DCR   B             Decr. byte count
C0E5 C2DDC0             JNZ   :C0DD         Next byte if not ready
C0E8 35                 DCR   M             Decr. hibyte
C0E9 7E                 MOV   A,M
C0EA FE7F               CPI   :7F           Check for overflow
C0EC CC4BC0             CZ    :C04B         Then run overflow error
C0EF E1         LC33    POP   H             Normal return
C0F0 C1                 POP   B
C0F1 F1                 POP   PSW
C0F2 C9                 RET
                        *
                        ************************************************
                        * INCREMENT FLOATING POINT NUMBER IN MEMORY *
                        ************************************************
                        *
                        * If number = 0, or exponent < 0, a '1' is added
                        * to the lsb of the mantissa. Else, the position
                        * of the least significant '1' is looked up, and
                        * a '1' is added to this position.
                        * If the lsb of the mantissa is already a rounded
                        * value, no increment occurs.
                        *
                        * Entry: HL points to 1st byte of FPT number.
                        * Exit:  All registers preserved.
                        *
C0F3 F5         FINM    PUSH  PSW
C0F4 C5                 PUSH  B
C0F5 D5                 PUSH  D
C0F6 E5                 PUSH  H
C0F7 1162C4             LXI   D,:C462       Addr FPT(1)
C0FA 7E                 MOV   A,M           Get exp.byte
C0FB E67F               ANI   :7F           Mask sign bit
C0FD CAACC1             JZ    :C1AC         If nr=0: add 1, abort
C100 FE40               CPI   :40           Is exponent negative?
C102 D2ACC1             JNC   :C1AC         Then add 1, abort
C105 FE19               CPI   :19           Lsb of mantissa is not lsb
                                            of number ?
C107 D24DC1             JNC   :C14D         Then Popall, ret
C10A BE                 CMP   M             Check if nr. is negative
C10B 23                 INX   H
C10C C252C1             JNZ   :C152         Then jump
```

```
                        * From LC34 also used by XFDCM.
                        * Find lsb of mantissa if nr is positive:
                        
C10F D609       LC34    SUI   :09       In 1st byte ?
C111 DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C114 DA36C1             JC    :C136     and jump
C117 23                 INX   H
C118 D608               SUI   :08       In 2nd byte ?
C11A DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C11D DA2EC1             JC    :C12E     and jump
C120 23                 INX   H
C121 D608               SUI   :08       In 3rd byte ?
C123 CDEEC1             CALL  :C1EE     Then SHL bit into A (A) time
C126 86                 ADD   M
C127 77                 MOV   M,A       Add 1 to 3rd byte mantissa
C128 D24DC1             JNC   :C14D     Ready if no overflow
C12B 2B                 DCX   H
C12C 3E01               MVI   A,:01     Overflow: add 1 to 2nd byte
C12E 86         LC35    ADD   M
C12F 77                 MOV   M,A       Add 1 to 2nd byte mantissa
C130 D24DC1             JNC   :C14D     Ready if no overflow
C133 2B                 DCX   H
C134 3E01               MVI   A,:01     Overflow: add 1 to 1st byte
C136 86         LC36    ADD   M
C137 77                 MOV   M,A       Add 1 to 1st byte mantissa
C138 D24DC1             JNC   :C14D     Ready if no overflow

                        * If overflow into exponent byte:

C13B 1F                 RAR             )
C13C 77                 MOV   M,A       )
C13D 23                 INX   H         )
C13E 7E                 MOV   A,M       ) Shift all bits in
C13F 1F                 RAR             ) mantissa right
C140 77                 MOV   M,A       ) one position
C141 23                 INX   H         )
C142 7E                 MOV   A,M       )
C143 1F                 RAR             )
C144 77                 MOV   M,A       )
C145 2B                 DCX   H
C146 2B                 DCX   H
C147 2B                 DCX   H         HL pnts to exp.byte
C148 3E01               MVI   A,:01
C14A CDBAC1     LC37    CALL  :C1BA     Add 1 to exponent
                        *
C14D E1         EXIT    POP   H
C14E D1                 POP   D
C14F C1                 POP   B
C150 F1                 POP   PSW
C151 C9                 RET

                        * Find lsb of mantissa if nr is negative:

C152 D609       LC39    SUI   :09       In 1st byte ?
C154 DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C157 DA7DC1             JC    :C17D     and jump
C15A 23                 INX   H
C15B D608               SUI   :08       In 2nd byte ?
C15D DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C160 DA73C1             JC    :C173     and jump
C163 23                 INX   H
C164 D608               SUI   :08       In 3rd byte ?
```

```
374 C166 DCEEC1              CC    :C1EE     Then SHL bit into A (A) time
375 C169 47                  MOV   B,A
376 C16A 7E                  MOV   A,M
377 C16B 90                  SUB   B         Subtract 1 from 3rd byte
378 C16C 77                  MOV   M,A
379 C16D D24DC1              JNC   :C14D     Ready if no borrow
380 C170 2B                  DCX   H
381 C171 3E01                MVI   A,:01     Subtract 1 from 2nd byte if
382                                          borrow
383 C173 47          LC40    MOV   B,A
384 C174 7E                  MOV   A,M
385 C175 90                  SUB   B         Subtract 1 from 2nd byte
386 C176 77                  MOV   M,A
387 C177 D24DC1              JNC   :C14D     Ready if no borrow
388 C17A 2B                  DCX   H
389 C17B 3E01                MVI   A,:01     Subtract 1 from 1st byte if
390                                          borrow
391 C17D 47          LC41    MOV   B,A
392 C17E 7E                  MOV   A,M
393 C17F 90                  SUB   B         Subtract 1 from 1st byte
394 C180 77                  MOV   M,A
395 C181 FA4DC1              JM    :C14D     Ready if normalised
396
397                  * If not normalised:
398
399 C184 061B                MVI   B,:18     Nr of mantissa bits
400 C186 23          LC42    INX   H         )
401 C187 23                  INX   H         )
402 C188 B7                  ORA   A         )
403 C189 7E                  MOV   A,M       )
404 C18A 17                  RAL             )
405 C18B 77                  MOV   M,A       ) Shift all bits
406 C18C 2B                  DCX   H         ) of mantissa
407 C18D 7E                  MOV   A,M       ) left one position
408 C18E 17                  RAL             )
409 C18F 77                  MOV   M,A       )
410 C190 2B                  DCX   H         )
411 C191 7E                  MOV   A,M       )
412 C192 17                  RAL             )
413 C193 77                  MOV   M,A       )
414 C194 B7                  ORA   A
415 C195 FA9FC1              JM    :C19F     If normalized
416 C198 05                  DCR   B         Update exp. count
417 C199 CAA6C1              JZ    :C1A6     If exp. now zero
418 C19C C386C1              JMP   :C186     Cont. normalisation
419
420                  * Normalisation done:
421
422 C19F 2B          LC43    DCX   H         Pnts to exp.byte
423 C1A0 78                  MOV   A,B       Get exp. count
424 C1A1 D619                SUI   :19       Minus nr of bytes in
425                                          mantissa
426 C1A3 C34AC1              JMP   :C14A     Update exponent, quit
427
428                  * If exponent is zero:
429
430 C1A6 2B          LC44    DCX   H
431 C1A7 3600                MVI   M,:00     Exp.byte is 0
432 C1A9 C34DC1              JMP   :C14D     Popall, ret
433
434                  * Simply add 1 (FINM) or add -1 (FDCM):
435
```

```
C1AC E7          LC45    RST   4          Copy number into MACC
C1AD 0C                  DATA  :0C
C1AE EB                  XCHG
C1AF E7                  RST   4          Add 1 or -1 (FPT)
C1B0 00                  DATA  :00
C1B1 EB                  XCHG
C1B2 E7                  RST   4          Copy MACC into memory
C1B3 0F                  DATA  :0F
C1B4 C34DC1              JMP   :C14D      Popall, ret
                 *
                 ******************
                 * ADD EXPONENTS *
                 ******************
                 *
                 * LC225 for operand in MACC.
                 * LC46 for operand in M.
                 *
                 * Entry: Byte to be added to exponent in A.
                 * Exit:  BCDEHL preserved.
                 *        CY=0: O.K.
                 *        CY=1: Overflow.
                 *
C1B7 21D500      LC225   LXI   H,:00D5    Addr. MACC
                 *
C1BA E5          LC46    PUSH  H
C1BB D5                  PUSH  D
C1BC C5                  PUSH  B
C1BD 4F                  MOV   C,A        Byte to be added in C
C1BE 7E                  MOV   A,M        Get exp.byte operand
C1BF E680                ANI   :80        Sign bit mantissa only
C1C1 47                  MOV   B,A        in B
C1C2 7E                  MOV   A,M        Get exp.byte
C1C3 CDE9C1              CALL  :C1E9      Sign extend
C1C6 F5                  PUSH  PSW        Save sign extended exp.byte
C1C7 A9                  XRA   C          XOR with byte to be added
C1C8 2F                  CMA
C1C9 57                  MOV   D,A
C1CA F1                  POP   PSW        Get sign extended exp.byte
C1CB 81                  ADD   C          Add byte to exponent
C1CC 4F                  MOV   C,A        Store result
C1CD 1F                  RAR
C1CE A9                  XRA   C
C1CF A2                  ANA   D
C1D0 FAE2C1              JM    :C1E2      If overflow into sign bit
C1D3 79                  MOV   A,C        Get new exp.byte
C1D4 17                  RAL
C1D5 A9                  XRA   C
C1D6 FAE2C1              JM    :C1E2      If overflow into sign bit
C1D9 79                  MOV   A,C        Get new exp.byte
C1DA E67F                ANI   :7F        Exponent only
C1DC B0                  ORA   B          Add sign bit mantissa
C1DD 77                  MOV   M,A        Store it
C1DE C1          LC47    POP   B
C1DF D1                  POP   D
C1E0 E1                  POP   H
C1E1 C9                  RET              CY=0

                 * If overflow into sign bit:

C1E2 79          LC48    MOV   A,C        Get new exp.byte
C1E3 17                  RAL
C1E4 B7                  ORA   A
```

```
498 C1E5 37                 STC
499 C1E6 C3DEC1             JMP   :C1DE     Abort with CY=1
500                 *
501                 ***************
502                 * SIGN EXTEND *
503                 ***************
504                 *
505                 * Exponent byte is normalized.
506                 *
507                 * Entry: Exp.byte in A.
508                 * Exit:  BCDEHL preserved.
509                 *        Normalized exp.byte in A: Exp.value in
510                 *        bits 7-2, sign mantissa in bit 1, sign
511                 *        exponent in bit 0.
512                 *
513 C1E9 07        SEXT    RLC
514 C1EA 07                RLC
515 C1EB 07                RLC
516 C1EC 1F                RAR
517 C1ED C9                RET
518                 *
519                 ************************************
520                 * MOVE A BIT INTO A LEFT (A) TIMES *
521                 ************************************
522                 *
523                 * Used to place a '1' in the correct position in
524                 * a byte for adding/subtracting '1' to/from the
525                 * least significant '1' of a FPT mantissa.
526                 *
527                 * Entry: A contains a neg. number indicating
528                 *        how often RAL has to be performed.
529                 * Exit:  Result in A,B.
530                 *        FCDEHL preserved.
531                 *
532 C1EE F5        LC50    PUSH  PSW
533 C1EF 47                MOV   B,A       Save nr of shifts
534 C1F0 AF                XRA   A         Clear A
535 C1F1 37                STC             Set CY
536 C1F2 17        LC51    RAL             SHL
537 C1F3 04                INR   B         Update count
538 C1F4 C2F2C1            JNZ   :C1F2     Continu if not ready
539 C1F7 47                MOV   B,A       Save result
540 C1F8 F1                POP   PSW
541 C1F9 78                MOV   A,B       Result in A
542 C1FA C9                RET
543                 *
544                 *
545                 *
546 C1FB                   END
```

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

BASE   C000   DECBUF C033   EXIT   C14D   FCOMP  C079
FINM   C0F3   FPEAE  C05E   FPEDO  C06C   FPEOV  C04B
FPEUN  C065   ICOMP  C0AC   IDCM   C0D5   IINM   C0BB
LC18   C047   LC20   C04F   LC22   C073   LC225  C1B7
LC23   C087   LC24   C08B   LC241  C0B7   LC25   C09F
LC26   C0A2   LC27   C0A3   LC28   C0A4   LC29   C0A6
LC30   C0C2   LC31   C0D2   LC32   C0DD   LC33   C0EF
LC34   C10F   LC35   C12E   LC36   C136   LC37   C14A
```

LC39	C152	LC40	C173	LC41	C17D	LC42	C186
LC43	C19F	LC44	C1A6	LC45	C1AC	LC46	C1BA
LC47	C1DE	LC48	C1E2	LC50	C1EE	LC51	C1F2
MINIT	C035	SEXT	C1E9	XFBC	C021	XFCB	C01E
XFCOMP	C00C	XFDCM	C009	XFINM	C006	XHBC	C02D
XHCB	C02A	XIBC	C027	XICB	C024	XICOMP	C015
XIDCM	C012	XIINM	C00F	XINIT	C003	XPOP	C01B
XPRTY	C030	XPUSH	C018				

```
002                             ORG   :C1FB
003                     *
004                     *
005                     *
006                     ***********************************************
007                     * DECREMENT FLOATING POINT NUMBER IN MEMORY *
008                     ***********************************************
009                     *
010                     * Routine is not used.
011                     *
012                     * If the number is 0, or the exponent < 0, -1 is
013                     * added to the mantissa. Else, a -1 is added/
014                     * subtracted to/from the least significant '1' of
015                     * the mantissa.
016                     * If the lsb of the mantissa is already a rounded
017                     * value, no decrement occurs.
018                     *
019                     * Entry: HL points to FPT number in M.
020                     * Exit:  All registers preserved.
021                     *
022 C1FB F5             FDCM    PUSH  PSW
023 C1FC C5                     PUSH  B
024 C1FD D5                     PUSH  D
025 C1FE E5                     PUSH  H
026 C1FF 111AC2                 LXI   D,:C21A     Addr. FPT(-1)
027 C202 7E                     MOV   A,M         Get exp.byte
028 C203 E67F                   ANI   :7F         Mask sign bit
029 C205 CAACC1                 JZ    :C1AC       If nr=0: add -1, abort
030 C208 FE40                   CPI   :40         Is exp. negative ?
031 C20A D2ACC1                 JNC   :C1AC       Then add -1, abort
032 C20D FE18                   CPI   :18         Max. nr of mantissa bits
033 C20F D24DC1                 JNC   :C14D       Abort if lsb mantissa is
034                                               not lsb of number
035 C212 BE                     CMP   M           Check if nr. is negative
036 C213 23                     INX   H
037 C214 CA52C1                 JZ    :C152       Into FINM for neg. nr
038 C217 C30FC1                 JMP   :C10F       Idem for pos. nr.
039                     *
040                     * DATA - (not used):
041                     *
042 C21A 81             FPM1    DATA  :81         FPT (-1)
043 C21B 80                     DATA  :80
044 C21C 00                     DATA  :00
045 C21D 00                     DATA  :00
046                     *
047                     ************************
048                     * SAVE MACC ON STACK *
049                     ************************
050                     *
051                     * Contents MACC is placed on TOS. Returnaddress
052                     * is saved.
053                     *
054                     * Entry: None.
055                     * Exit:  All registers preserved.
056                     *        On stack: HL; returnaddress; MACC.
057                     *
058 C21E 22E100         PUSH    SHLD  :00E1       Save HL
059 C221 E3                     XTHL              Get returnaddress
060 C222 22DF00                 SHLD  :00DF       Save it
061 C225 E5                     PUSH  H           and put it on stack again
062 C226 210000                 LXI   H,:0000
063 C229 39                     DAD   SP          SP in HL
```

```
064 C22A E7              RST   4          Copy MACC to TOS
065 C22B 0F              DATA  :0F
066 C22C 2ADF00          LHLD  :00DF      Get returnaddr.
067 C22F E5              PUSH  H          on stack
068 C230 2AE100    LC52  LHLD  :00E1      Get original HL
069 C233 C9              RET
070                *
071                ******************************
072                * RETRIEVE MACC FROM STACK *
073                ******************************
074                *
075                * Gets data from TOS and place it in MACC.
076                *
077                * Entry: None.
078                * Exit:  All registers preserved.
079                *
080 C234 22E100    POP   SHLD  :00E1      Save HL
081 C237 E1              POP   H          Get returnaddress
082 C238 22DF00          SHLD  :00DF      Save it
083 C23B 210000          LXI   H,:0000
084 C23E 39              DAD   SP         Get SP in HL
085 C23F E7              RST   4          Copy TOS to MACC
086 C240 0C              DATA  :0C
087 C241 E1              POP   H
088 C242 2ADF00          LHLD  :00DF      Get returnaddress
089 C245 E3              XTHL             on stack
090 C246 C330C2          JMP   :C230      Restore HL, ret
091                *
092                ******************************************
093                * INPUT A FLOATING POINT NUMBER TO MACC *
094                ******************************************
095                *
096                * Converts a FPT number to binary into MACC.
097                * The input string is converted as a integer FPT
098                * number, then multiplied/divided by a power of
099                * 10, corresponding to the explicit exponent and
100                * placement of the decimal point.
101                *
102                * Entry: C points to 1st digit of FPT nr in input.
103                * Exit:  CY=1: No error.
104                *        CY=0: Over/underflow error.
105                *        C points past FPT string in input.
106                *        ABDEHL, rest of F preserved.
107                *
108 C249 37        FCB   STC              CY=1
109 C24A F5              PUSH  PSW
110 C24B D5              PUSH  D
111 C24C E5              PUSH  H
112 C24D CDAEC2          CALL  :C2AE      Clear MACC+DEH; L=2B
113 C250 CD2FC3    LC53  CALL  :C32F      Get bin.value of input char
114 C253 DCBAC2          CC    :C2BA      Value found: Move digit into
115                                       MACC
116 C256 DA50C2          JC    :C250      and get next digit
117 C259 FE2E            CPI   :2E        '.' ?
118 C25B CA6BC2          JZ    :C26B      Then jump
119 C25E 1D              DCR   E
120 C25F 1C              INR   E
121 C260 CAA6C2          JZ    :C2A6      If error
122 C263 FE45            CPI   :45        'E' ?
123 C265 CA84C2          JZ    :C284      Then jump
124 C268 C39FC2          JMP   :C29F      Convert FPT exp; quit
125
```

```
126                     * If digit is '.':
127
128 C26B CDD5C2         LC54    CALL    :C2D5       E=0, H=H+1
129 C26E CD2FC3         LC55    CALL    :C32F       Get bin.value of input char
130 C271 DCBAC2                 CC      :C2BA       If found: Move digit into
131                                                 MACC
132 C274 DA6EC2                 JC      :C26E       and get next digit
133 C277 1D                     DCR     E
134 C278 1C                     INR     E
135 C279 CAA6C2                 JZ      :C2A6       If error
136 C27C FE45                   CPI     :45         'E' ?
137 C27E C4D9C2                 CNZ     :C2D9       H=0 if not
138 C281 C29FC2                 JNZ     :C29F       If not: convert exp, quit
139
140                     * If digit is 'E':
141
142 C284 CDD9C2         LC56    CALL    :C2D9       H=0
143 C287 CD2FC3                 CALL    :C32F       Get bin.value of input char
144 C28A CCDCC2                 CZ      :C2DC       If '+' or '-': char in L
145 C28D CC2FC3                 CZ      :C32F       Then get bin.value of next
146                                                 char
147 C290 D2A6C2                 JNC     :C2A6       Error if no char found
148 C293 CDDEC2                 CALL    :C2DE       H = 10 * H + A
149 C296 CD2FC3                 CALL    :C32F       Get bin.value of input char
150 C299 DCDEC2                 CC      :C2DE       If found: H = 10 * H + A
151 C29C DC2FC3                 CC      :C32F       Get bin.value of input char
152
153                     * If digit is number:
154
155 C29F CDEBC2         LC57    CALL    :C2EB       Convert FPT exponent
156 C2A2 E1             LC58    POP     H
157 C2A3 D1                     POP     D
158 C2A4 F1                     POP     PSW         CY=1
159 C2A5 C9                     RET
160
161                     * If error:
162
163 C2A6 CD2DC3         LC59    CALL    :C32D       DCR C
164 C2A9 E1             LC60    POP     H
165 C2AA D1                     POP     D
166 C2AB F1                     POP     PSW
167 C2AC 3F                     CMC                 CY=0
168 C2AD C9                     RET
169                     *
170                     * CLEAR MACC AND REGISTERS D, E AND H:
171                     *
172                     * Exit:  ABC preserved.
173                     *        L = 2B ('+')
174                     *
175 C2AE 215EC4         LC61    LXI     H,:C45E     Addr. FPT(0)
176 C2B1 E7                     RST     4           Copy FPT(0) to MACC
177 C2B2 0C                     DATA    :0C
178 C2B3 110000                 LXI     D,:0000     Clear DE
179 C2B6 212B00                 LXI     H,:002B     Clear H, L='+'
180 C2B9 C9                     RET
181                     *
182                     * MOVE A DIGIT INTO THE MACC:
183                     *
184                     * MACC = MACC * 10 + A.
185                     *
186                     * Entry: A: Digit 1 - 9.
187                     * Exit:  AFBCHL preserved.
```

```
188                     *       D=D-H; E=E-1.
189                     *
190 C2BA F5             LC62    PUSH  PSW
191 C2BB E5                     PUSH  H
192 C2BC 214DC3                 LXI   H,:C34D   Addr FPT (10)
193 C2BF E7                     RST   4         MACC = MACC * 10 (FPT)
194 C2C0 06                     DATA  :06
195 C2C1 D5                     PUSH  D
196 C2C2 87                     ADD   A         )
197 C2C3 87                     ADD   A         ) DE = 4 * A
198 C2C4 5F                     MOV   E,A       ) (calc offset to startaddr)
199 C2C5 1600                   MVI   D,:00     )
200 C2C7 215EC4                 LXI   H,:C45E   Addr table FPT(1-9)
201 C2CA 19                     DAD   D         Calc. addr nr to be added
202 C2CB D1                     POP   D
203 C2CC E7                     RST   4         MACC = MACC + (1-9) (FPT)
204 C2CD 00                     DATA  :00
205 C2CE E1                     POP   H
206 C2CF 7A                     MOV   A,D
207 C2D0 94                     SUB   H
208 C2D1 57                     MOV   D,A       D=D-H
209 C2D2 1D                     DCR   E         E=E-1
210 C2D3 F1                     POP   PSW
211 C2D4 C9                     RET
212                     *
213 C2D5 1E00           LC63    MVI   E,:00
214 C2D7 24                     INR   H
215 C2D8 C9                     RET
216                     *
217 C2D9 2600           LC64    MVI   H,:00
218 C2DB C9                     RET
219                     *
220 C2DC 6F             LC65    MOV   L,A
221 C2DD C9                     RET
222                     *
223                     * H = 10 * H + A:
224                     *
225                     * Exit: AFBCDEL preserved.
226                     *
227 C2DE F5             LC66    PUSH  PSW
228 C2DF 7C                     MOV   A,H       A=H
229 C2E0 87                     ADD   A         A=2*H
230 C2E1 87                     ADD   A         A=4*H
231 C2E2 84                     ADD   H         A=5*H
232 C2E3 87                     ADD   A         A=10*H
233 C2E4 67                     MOV   H,A
234 C2E5 F1                     POP   PSW
235 C2E6 F5                     PUSH  PSW
236 C2E7 84                     ADD   H         A=10*H+A
237 C2E8 67                     MOV   H,A
238 C2E9 F1                     POP   PSW
239 C2EA C9                     RET
240                     *
241                     * CONVERT A FPT EXPONENT:
242                     *
243                     * The MACC is multiplied/divided by a power of 10
244                     * corresponding to the 'E'-exponent minus the number
245                     * of digits after the decimal point.
246                     *
247                     * Entry: C:    Points beyond 1st nonuseable char of
248                     *              FPT number in input.
249                     *       L:     Contains sign of exponent.
```

```
250                     *         H:    Contains 'E....' exponent (10).
251                     *         MACC: Contains FPT conversion of string of
252                     *               digits.
253                     *         D:    Contains nr of digits after '.'.
254                     * Exit:   BE preserved, AHL corrupted.
255                     *         C:    Decremented to after FPT nr in input.
256                     *         D:    Contains effective exponent.
257                     *
258 C2EB CD2DC3      LC67    CALL  :C32D       Decr C
259 C2EE 7D                  MOV   A,L         Get exp. sign
260 C2EF FE2D                CPI   :2D         '-' ?
261 C2F1 7C                  MOV   A,H         Get exponent
262 C2F2 C2F7C2              JNZ   :C2F7       If exp. positive
263 C2F5 2F                  CMA               ) Else: make exponent
264 C2F6 3C                  INR   A           ) positive
265 C2F7 82          LC68    ADD   D           Add nr of digits after '.'
266 C2F8 57                  MOV   D,A         Save result
267 C2F9 F2FEC2              JP    :C2FE       If result positive
268 C2FC 2F                  CMA               ) Else: make it
269 C2FD 3C                  INR   A           ) positive
270 C2FE C5          LC69    PUSH  B
271 C2FF 0605                MVI   B,:05       Nr of times of multipl.
272 C301 214DC3              LXI   H,:C34D     Addr table powers of 10
273 C304 B7          LC70    ORA   A           Flags on result LC68
274 C305 1F                  RAR               lsb in carry
275 C306 D217C3              JNC   :C317       If bit=0
276 C309 F5                  PUSH  PSW
277 C30A 7A                  MOV   A,D         Check if multipl/div
278 C30B B7                  ORA   A
279 C30C FA11C3              JM    :C311       If division
280 C30F E7                  RST   4           MACC = MACC * power of 10
281 C310 06                  DATA  :06
282 C311 CA16C3      LC71    JZ    :C316       If multiplication
283 C314 E7                  RST   4           MACC = MACC / power of 10
284 C315 09                  DATA  :09
285 C316 F1          LC72    POP   PSW         Restore A
286 C317 23          LC73    INX   H
287 C318 23                  INX   H
288 C319 23                  INX   H
289 C31A 23                  INX   H           HL pnts to next ^10
290 C31B 05                  DCR   B
291 C31C C204C3              JNZ   :C304       Again if not ready
292 C31F B7                  ORA   A
293 C320 CA2BC3              JZ    :C32B       If result OK
294
295                     * If error:
296
297 C323 7A                  MOV   A,D
298 C324 B7                  ORA   A           Set flags for error type
299 C325 F44BC0              CP    :C04B       If overflow error
300 C328 FC65C0              CM    :C065       If underflow error
301 C32B C1          LC74    POP   B           Normal return
302 C32C C9                  RET
303                     *
304 C32D 0D          LC75    DCR   C
305 C32E C9                  RET
306                     *
307                     * GET BINARY VALUE OF INPUT CHARACTER IN A:
308                     *
309                     * Entry: C points to character in input.
310                     * Exit:  C points to next character.
311                     *        BDEHL preserved.
```

```
                        *       CY=1, Z=0: Value in A.
                        *       CY=0, Z=1: Char is +/-.
                        *       CY=0, Z=0: otherwise.
                        *
C32F CD73C0             LC76    CALL  :C073     Get char from line
C332 0C                         INR   C         Update pointer
C333 FE2B                       CPI   :2B
C335 C8                         RZ              Abort if '+'
C336 FE2D                       CPI   :2D
C338 C8                         RZ              Abort if '-'
C339 FE30                       CPI   :30
C33B 3F                         CMC
C33C D0                         RNC             Abort if < #30
C33D FE3A                       CPI   :3A
C33F D24BC3                     JNC   :C34B     Abort if > #3A
C342 D630                       SUI   :30       Convert ASCII to binary
C344 D5                         PUSH  D
C345 57                         MOV   D,A
C346 3C                         INR   A         Set Z-flag correctly for
                                                reqd output
C347 7A                         MOV   A,D
C348 D1                         POP   D
C349 37                         STC             CY=1: value in A
C34A C9                         RET
C34B B7                 LC77    ORA   A         Set flags correctly
C34C C9                         RET
                        *
                        ****************************
                        * TABLE FPT POWERS OF 10 *
                        ****************************
                        *
C34D 04                 LC233   DATA  :04       FPT 10^1
C34E A0                         DATA  :A0
C34F 00                         DATA  :00
C350 00                         DATA  :00
                        *
C351 07                         DATA  :07       FPT 10^2
C352 C8                         DATA  :C8
C353 00                         DATA  :00
C354 00                         DATA  :00
                        *
C355 0E                         DATA  :0E       FPT 10^4
C356 9C                         DATA  :9C
C357 40                         DATA  :40
C358 00                         DATA  :00
                        *
C359 1B                         DATA  :1B       FPT 10^8
C35A BE                         DATA  :BE
C35B BC                         DATA  :BC
C35C 20                         DATA  :20
                        *
C35D 36                         DATA  :36       FPT 10^16
C35E 8E                         DATA  :8E
C35F 1B                         DATA  :1B
C360 CA                         DATA  :CA
                        *
                        ************************************************
                        * CONVERT A FLOATING POINT NUMBER FOR OUTPUT *
                        ************************************************
                        *
                        * A FPT number in MACC is converted to ASCII in
                        * outputbuffer 00E4-F1. The sign is in 00E4, the
```

```
374                     * decimal point in 00E5. The normalized value of
375                     * the mantissa is in 00E6-EC (7 digits). In 00F1
376                     * is the 10's exponent in 2-complement binairy
377                     * signed format.
378                     *
379                     * Exit: A=6 (number of significant digits).
380                     *       BCDEHL, MACC preserved.
381                     *
382 C361 C5            FBC     PUSH  B
383 C362 D5                    PUSH  D
384 C363 E5                    PUSH  H
385 C364 CD1EC2                CALL  :C21E     Save MACC on stack
386 C367 E7                    RST   4         Copy MACC to reg A,B,C,D
387 C368 15                    DATA  :15
388 C369 F5                    PUSH  PSW       Save exp.byte
389 C36A B0                    ORA   B
390 C36B B1                    ORA   C
391 C36C B2                    ORA   D
392 C36D CAD4C3                JZ    :C3D4     If FPT nr is zero
393 C370 F1                    POP   PSW       Get exp.byte
394 C371 F5                    PUSH  PSW
395 C372 2600                  MVI   H,:00
396 C374 E67F                  ANI   :7F       Mask sign bit mantissa
397 C376 FE40                  CPI   :40
398 C378 DA80C3                JC    :C380     If exp is positive
399 C37B 25                    DCR   H
400 C37C 2F                    CMA             ) Else: convert
401 C37D E67F                  ANI   :7F       ) exponent to
402 C37F 3C                    INR   A         ) positive
403 C380 F5            LC78    PUSH  PSW       Save value exponent
404 C381 AF                    XRA   A
405 C382 E7                    RST   4         Copy mantissa to MACC
406 C383 12                    DATA  :12
407 C384 F1                    POP   PSW       Get exp.value
408 C385 44                    MOV   B,H       B=FF (exp.<0), 00 (exp.>0)
409 C386 2137C4                LXI   H,:C437   Addr table powers FPT(2)
410 C389 0E00                  MVI   C,:00
411 C38B 1607                  MVI   D,:07     7 digits to be examined
412 C38D 0F            LC79    RRC             Shift exp. into carry
413 C38E F5                    PUSH  PSW       Save rest of exp.
414 C38F 7E                    MOV   A,M       Get 10's power byte
415 C390 23                    INX   H         Points to next
416 C391 D2A2C3                JNC   :C3A2     If n-th power of 2=0:
417                                            go to next
418 C394 81                    ADD   C
419 C395 4F                    MOV   C,A       Total 10's power in C
420 C396 05                    DCR   B
421 C397 04                    INR   B
422 C398 C29DC3                JNZ   :C39D     Jump if exp. negative
423 C39B E7                    RST   4         Multipl. mantissa by
424 C39C 06                    DATA  :06       (2^2^n)/10^m
425 C39D CAA2C3        LC80    JZ    :C3A2
426 C3A0 E7                    RST   4         Divide mantissa by
427 C3A1 09                    DATA  :09       (2^2^n)/10^m
428 C3A2 23            LC81    INX   H
429 C3A3 23                    INX   H
430 C3A4 23                    INX   H
431 C3A5 23                    INX   H         Pnts to next in table
432 C3A6 F1                    POP   PSW       Get rest exponent
433 C3A7 15                    DCR   D         Decr digit count
434 C3A8 C28DC3                JNZ   :C38D     Again if not 7 digits done
435 C3AB 05                    DCR   B
```

```
436 C3AC 04                 INR   B
437 C3AD 215AC4             LXI   H,:C45A   Addr FPT (0.1)
438 C3B0 C2C4C3             JNZ   :C3C4     If exp. negative
439
440                 * If exponent positive:
441
442 C3B3 E5        LC82    PUSH  H
443 C3B4 2162C4             LXI   H,:C462   Addr FPT(1)
444 C3B7 CD79C0             CALL  :C079     Compare with 1
445 C3BA E1                 POP   H
446 C3BB FAD4C3             JM    :C3D4     Jump if normalized
447 C3BE E7                 RST   4         MACC = MACC * 0.1 (FPT)
448 C3BF 06                 DATA  :06
449 C3C0 0C                 INR   C         Update 10's power
450 C3C1 C3B3C3             JMP   :C3B3     Cont. normalisation
451
452                 * If exponent negative:
453
454 C3C4 79        LC83    MOV   A,C       )
455 C3C5 2F                 CMA             ) Change 10's power
456 C3C6 3C                 INR   A         ) to neg. value
457 C3C7 4F                 MOV   C,A       )
458 C3C8 CD79C0    LC84    CALL  :C079     Compare with 0,1
459 C3CB F2D4C3             JP    :C3D4     Jump if normalized
460 C3CE E7                 RST   4         MACC = MACC / 0.1 (FPT)
461 C3CF 09                 DATA  :09
462 C3D0 0D                 DCR   C         Update 10's power
463 C3D1 C3C8C3             JMP   :C3C8     Cont. normalisation
464
465                 * Load output buffer:
466
467 C3D4 79        LC85    MOV   A,C       Get 10's power
468 C3D5 32F100              STA   :00F1     In output buffer
469 C3D8 F1                 POP   PSW       Get signbyte mantissa
470 C3D9 B7                 ORA   A         Set flags on it
471 C3DA 21E400             LXI   H,:00E4   Addr output buffer
472 C3DD 362B               MVI   M,:2B     '+' in buffer
473 C3DF F2E4C3             JP    :C3E4     If mantissa is positive
474 C3E2 362D               MVI   M,:2D     Else: '-' in buffer
475 C3E4 23        LC86    INX   H
476 C3E5 362E               MVI   M,:2E     '.' in 00E5
477 C3E7 23                 INX   H
478 C3E8 E5                 PUSH  H
479 C3E9 E7                 RST   4         Copy MACC to reg A,B,C,D
480 C3EA 15                 DATA  :15
481 C3EB F5                 PUSH  PSW       Save exp.byte
482 C3EC AF                 XRA   A
483 C3ED E7                 RST   4         Copy mantissa to MACC
484 C3EE 12                 DATA  :12
485 C3EF F1                 POP   PSW       Get exp.byte
486 C3F0 2F                 CMA             )
487 C3F1 3C                 INR   A         ) 2-compl.
488 C3F2 E67F               ANI   :7F       Mask sign bit mantissa
489 C3F4 2110C6             LXI   H,:C610   Addr INT(10)
490 C3F7 E7                 RST   4         MACC = MACC * 10 (INT)
491 C3F8 54                 DATA  :54
492 C3F9 2120C4             LXI   H,:C420   Addr. INT(1)
493 C3FC 3D                 DCR   A
494 C3FD FA02C4             JM    :C402     If exp. converted
495 C400 E7                 RST   4         Shift MACC right
496 C401 72                 DATA  :72
497 C402 F2FCC3    LC88     JP    :C3FC     If not ready
```

```
498 C405 E1                     POP   H
499 C406 E7                     RST   4          Copy MACC to reg A,B,C,D
500 C407 15                     DATA  :15
501 C408 C630                   ADI   :30        Exp.byte in ASCII
502 C40A 77                     MOV   M,A        Into outputbuffer
503 C40B 23                     INX   H
504 C40C 0606                   MVI   B,:06      6 sign. digits for mantissa
505 C40E CD24C4       LC89      CALL  :C424      Convert one digit to ASCII
506 C411 77                     MOV   M,A        Into output buffer
507 C412 23                     INX   H
508 C413 05                     DCR   B
509 C414 C20EC4                 JNZ   :C40E      Next digit if not ready
510 C417 CD34C2                 CALL  :C234      Retrieve MACC from TOS
511 C41A E1                     POP   H
512 C41B D1                     POP   D
513 C41C C1                     POP   B
514 C41D 3E06                   MVI   A,:06      6 sign. digits in outputbuf
515 C41F C9                     RET
516                   *
517                   * DATA:
518                   *
519 C420 00           I1        DATA  :00        INT (1)
520 C421 00                     DATA  :00
521 C422 00                     DATA  :00
522 C423 01                     DATA  :01
523                   *
524                   * CONVERT A DIGIT FOR OUTPUT:
525                   *
526                   * Highest byte of MACC *10 is made ASCII.
527                   *
528                   * Exit: A: Converted highest byte MACC.
529                   *       BCHL preserved. DE corrupted.
530                   *
531 C424 C5           LC90      PUSH  B
532 C425 E5                     PUSH  H
533 C426 E7                     RST   4          Copy MACC to reg A,B,C,D
534 C427 15                     DATA  :15
535 C428 AF                     XRA   A          Clear highest byte
536 C429 E7                     RST   4          Copy reg A,B,C,D to MACC
537 C42A 12                     DATA  :12
538 C42B 2110C6                 LXI   H,:C610    Addr INT(10)
539 C42E E7                     RST   4          MACC = MACC * 10 (INT)
540 C42F 54                     DATA  :54
541 C430 E7                     RST   4          Copy MACC to reg A,B,C,D
542 C431 15                     DATA  :15
543 C432 C630                   ADI   :30        Make highest byte ASCII
544 C434 E1                     POP   H
545 C435 C1                     POP   B
546 C436 C9                     RET
547                   *
548                   *
549                   *
550 C437                        END
```

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

FBC	C361	FCB	C249	FDCM	C1FB	FPM1	C21A
I1	C420	LC233	C34D	LC52	C230	LC53	C250
LC54	C26B	LC55	C26E	LC56	C284	LC57	C29F
LC58	C2A2	LC59	C2A6	LC60	C2A9	LC61	C2AE

LC62	C2BA	LC63	C2D5	LC64	C2D9	LC65	C2DC
LC66	C2DE	LC67	C2EB	LC68	C2F7	LC69	C2FE
LC70	C304	LC71	C311	LC72	C316	LC73	C317
LC74	C32B	LC75	C32D	LC76	C32F	LC77	C34B
LC78	C380	LC79	C38D	LC80	C39D	LC81	C3A2
LC82	C3B3	LC83	C3C4	LC84	C3C8	LC85	C3D4
LC86	C3E4	LC88	C402	LC89	C40E	LC90	C424
POP	C234	PUSH	C21E				

```
002                             ORG   :C437
003                     *
004                     *
005                     *
006                     ************************
007                     * FPT NUMBER CONSTANTS *
008                     ************************
009                     *
010                     * For the first 7 numbers, the 1st byte is the
011                     * power of 10 for division.
012                     *
013 C437 00             LC234   DATA  :00
014 C438 02                     DATA  :02       FPT (2^1)/10^0
015 C439 80                     DATA  :80
016 C43A 00                     DATA  :00
017 C43B 00                     DATA  :00
018                     *
019 C43C 00                     DATA  :00
020 C43D 03                     DATA  :03       FPT (2^2)/10^0
021 C43E 80                     DATA  :80
022 C43F 00                     DATA  :00
023 C440 00                     DATA  :00
024                     *
025 C441 01                     DATA  :01
026 C442 01                     DATA  :01       FPT (2^4)/10^1
027 C443 CC                     DATA  :CC
028 C444 CC                     DATA  :CC
029 C445 CD                     DATA  :CD
030                     *
031 C446 02                     DATA  :02
032 C447 02                     DATA  :02       FPT (2^8)/10^2
033 C448 A3                     DATA  :A3
034 C449 D7                     DATA  :D7
035 C44A 0A                     DATA  :0A
036                     *
037 C44B 04                     DATA  :04
038 C44C 03                     DATA  :03       FPT (2^16)/10^4
039 C44D D1                     DATA  :D1
040 C44E B7                     DATA  :B7
041 C44F 17                     DATA  :17
042                     *
043 C450 09                     DATA  :09
044 C451 03                     DATA  :03       FPT (2^32)/10^9
045 C452 89                     DATA  :89
046 C453 70                     DATA  :70
047 C454 5F                     DATA  :5F
048                     *
049 C455 13                     DATA  :13
050 C456 01                     DATA  :01       FPT (2^64)/10^19
051 C457 EC                     DATA  :EC
052 C458 1E                     DATA  :1E
053 C459 4A                     DATA  :4A
054                     *
055 C45A 7D             LC238   DATA  :7D       FPT (0.1)
056 C45B CC                     DATA  :CC
057 C45C CC                     DATA  :CC
058 C45D CD                     DATA  :CD
059                     *
060 C45E 00             FP0     DATA  :00       FPT (0)
061 C45F 00                     DATA  :00
062 C460 00                     DATA  :00
063 C461 00                     DATA  :00
```

```
064                     *
065 C462 01             FP1     DATA  :01       FPT (1)
066 C463 80                     DATA  :80
067 C464 00                     DATA  :00
068 C465 00                     DATA  :00
069                     *
070 C466 02             FP2     DATA  :02       FPT (2)
071 C467 80                     DATA  :80
072 C468 00                     DATA  :00
073 C469 00                     DATA  :00
074                     *
075 C46A 02             FP3     DATA  :02       FPT (3)
076 C46B C0                     DATA  :C0
077 C46C 00                     DATA  :00
078 C46D 00                     DATA  :00
079                     *
080 C46E 03             FP4     DATA  :03       FPT (4)
081 C46F 80                     DATA  :80
082 C470 00                     DATA  :00
083 C471 00                     DATA  :00
084                     *
085 C472 03             FP5     DATA  :03       FPT (5)
086 C473 A0                     DATA  :A0
087 C474 00                     DATA  :00
088 C475 00                     DATA  :00
089                     *
090 C476 03             FP6     DATA  :03       FPT (6)
091 C477 C0                     DATA  :C0
092 C478 00                     DATA  :00
093 C479 00                     DATA  :00
094                     *
095 C47A 03             FP7     DATA  :03       FPT (7)
096 C47B E0                     DATA  :E0
097 C47C 00                     DATA  :00
098 C47D 00                     DATA  :00
099                     *
100 C47E 04             FP8     DATA  :04       FPT (8)
101 C47F 80                     DATA  :80
102 C480 00                     DATA  :00
103 C481 00                     DATA  :00
104                     *
105 C482 04             FP9     DATA  :04       FPT (9)
106 C483 90                     DATA  :90
107 C484 00                     DATA  :00
108 C485 00                     DATA  :00
109                     *
110                     ************************************
111                     * PRETTIES UP FPT OR INT NUMBER *
112                     ************************************
113                     *
114                     * Entry: B:     Fix/float flag: (0=fix, 1=float).
115                     *        A:     Nr. of useable digits in string in
116                     *               00E4-F0 (not counting additional
117                     *               digit for rounding).
118                     *        00F1:  Nr. of digits before '.' (exponent).
119                     *        00E4:  Sign '+' or '-'.
120                     *        00E5:  Decimal point.
121                     *        E6-F0: Digits.
122                     * Exit:  All registers preserved.
123                     *        00E3:  Length of string.
124                     *        E4-F0: Output string.
125                     *
```

```
126                     * Format: Sign in 00E4 is blank or '-'.
127                     *         If exponent is 0:
128                     *              real case: '0.digits'
129                     *              int. case: no final '.'
130                     *              real case if INT: '.0'
131                     *         If exponent < -1:  E-format
132                     *         If exponent too large: E-format
133                     *         In E-format no '.0'
134                     *
135 C486 F5             PRTY    PUSH  PSW
136 C487 C5                     PUSH  B
137 C488 D5                     PUSH  D
138 C489 E5                     PUSH  H
139 C48A 21E600                 LXI   H,:00E6   Startaddr digits
140 C48D 4F                     MOV   C,A       Save nr. useable digits
141 C48E C5                     PUSH  B
142 C48F 0600                   MVI   B,:00
143 C491 09                     DAD   B         HL pnt to last useable digit
144 C492 7E                     MOV   A,M       Get last digit
145 C493 FE35                   CPI   :35       Check for rounding
146 C495 DAAFC4                 JC    :C4AF     If <5
147 C498 2B             LC91    DCX   H
148 C499 7E                     MOV   A,M       Get digit before
149 C49A FE39                   CPI   :39
150 C49C CAA3C4                 JZ    :C4A3     If it is 9
151 C49F 34                     INR   M         Rounding upwards
152 C4A0 C3AFC4                 JMP   :C4AF     Abort rounding
153 C4A3 3630           LC92    MVI   M,:30     Make digit before 0
154 C4A5 0D                     DCR   C         Decr. nr of digits
155 C4A6 C298C4                 JNZ   :C498     Cont. check for rounding
156 C4A9 3631                   MVI   M,:31     Make nr=1 if all digits 9
157 C4AB 21F100                 LXI   H,:00F1
158 C4AE 34                     INR   M         Incr nr of digits before '.'
159 C4AF C1             LC93    POP   B
160 C4B0 0C                     INR   C         Incr. nr useable digits
161 C4B1 3AF100                 LDA   :00F1     Get nr of digits before '.'
162 C4B4 B7                     ORA   A
163 C4B5 CAD8C4                 JZ    :C4D8     If 0
164 C4B8 FAE1C4                 JM    :C4E1     If too many digits
165 C4BB 80                     ADD   B         Add fix/float flag
166 C4BC B9                     CMP   C
167 C4BD D2E1C4                 JNC   :C4E1     If too many digits
168 C4C0 CD9CC6                 CALL  :C69C     Restore A, insert '.'
169                                             after number string
170 C4C3 79             LC94    MOV   A,C       Length string in A
171 C4C4 CD4BC5                 CALL  :C54B     Calc nr of digits for output
172 C4C7 3C             LC95    INR   A         Add 1
173 C4C8 21E300                 LXI   H,:00E3
174 C4CB 77                     MOV   M,A       String length in outbuf
175 C4CC 23                     INX   H
176 C4CD 7E                     MOV   A,M       Get sign
177 C4CE FE2B                   CPI   :2B       '+' ?
178 C4D0 C2D5C4                 JNZ   :C4D5     Then abort
179 C4D3 3620                   MVI   M,:20     Replace '+' by blank
180 C4D5 C34DC1         LC96    JMP   :C14D     Popall, ret
181
182                     * If format '0.digits':
183
184 C4D8 CD1AC5           LC97    CALL  :C51A     Move string right 1 pos.
185 C4DB 3630                   MVI   M,:30     Insert 0 in 00E5
186 C4DD 0C                     INR   C         Update nr of digits
187 C4DE C3C3C4                 JMP   :C4C3     Update string
```

```
188
189                     * If too many digits:
190
191 C4E1 3E01        LC98    MVI   A,:01
192 C4E3 CD31C5             CALL  :C531      Move string left 1 pos,
193                                           Insert '.' after string
194 C4E6 79                 MOV   A,C        Get nr of digits
195 C4E7 0600               MVI   B,:00
196 C4E9 CD4BC5             CALL  :C54B      Calc nr of digits for output
197 C4EC 47                 MOV   B,A        in B
198 C4ED 3AF100             LDA   :00F1      Get nr of digits before '.'
199 C4F0 3D                 DCR   A          Minus 1
200 C4F1 3645               MVI   M,:45      'E' in buf after last digit
201 C4F3 23                 INX   H
202 C4F4 04                 INR   B          Incr. nr of digits
203 C4F5 B7                 ORA   A          Flags on exponent
204 C4F6 F2FFC4             JP    :C4FF      If exp. positive
205
206                     * If exponent negative:
207
208 C4F9 362D               MVI   M,:2D      Store '-' in buffer
209 C4FB 23                 INX   H
210 C4FC 04                 INR   B          Incr nr of digits
211 C4FD 2F                 CMA
212 C4FE 3C                 INR   A          2-compl of exponent
213
214                     * Exponent to buffer:
215
216 C4FF 110A2F      LC99    LXI   D,:2F0A
217 C502 93          LC100   SUB   E          Exp.-10 (unit value)
218 C503 14                 INR   D          ASCII-count 10's-value
219 C504 D202C5             JNC   :C502      If rest exp. still >10
220 C507 C63A               ADI   :3A        Convert rest to Ascii
221 C509 5F                 MOV   E,A        in E
222 C50A 7A                 MOV   A,D        Get 10's value
223 C50B FE30               CPI   :30
224 C50D CA13C5             JZ    :C513      If exp. <10
225 C510 77                 MOV   M,A        10's value exp. in buf
226 C511 23                 INX   H
227 C512 04                 INR   B          Incr nr of digits
228 C513 73          LC101   MOV   M,E        Unit value exp. in buf
229 C514 23                 INX   H
230 C515 04                 INR   B          Incr nr of digits
231 C516 78                 MOV   A,B        into A
232 C517 C3C7C4             JMP   :C4C7      Prepare string for output
233                     *
234                     * MOVE STRING IN OUTPUTBUFFER RIGHT 1 POS.:
235                     *
236                     * The contents of 00E5-EF is moved up one
237                     * position to 00E6-F0.
238                     *
239                     * Entry: No conditions.
240                     * Exit:  ABCDEF preserved.
241                     *        HL points to 00E5.
242                     *
243 C51A F5          LC102   PUSH  PSW
244 C51B C5                 PUSH  B
245 C51C D5                 PUSH  D
246 C51D 21F000             LXI   H,:00F0    Highest destination address.
247 C520 54                 MOV   D,H
248 C521 5D                 MOV   E,L
249 C522 1B                 DCX   D          Highest source address.
```

```
C523 060B            MVI   B,:0B     Number of bytes.
C525 1A      LC103   LDAX  D         Get byte
C526 77              MOV   M,A       and move it.
C527 1B              DCX   D
C528 2B              DCX   H
C529 05              DCR   B
C52A C225C5          JNZ   :C525     Next byte if not ready
C52D D1              POP   D
C52E C1              POP   B
C52F F1              POP   PSW
C530 C9              RET
                *
                * MOVE STRING IN OUTPUTBUFFER LEFT 1 POS.:
                *
                * The string, beginning on 00E6, is moved one
                * memory location downwards. A '.' is inserted
                * after the string.
                *
                * Entry: A: number of bytes to be transferred.
                * Exit:  All registers preserved.
                *
C531 F5      LC104   PUSH  PSW
C532 C5              PUSH  B
C533 D5              PUSH  D
C534 E5              PUSH  H
C535 47              MOV   B,A       Store number of bytes
C536 21E500          LXI   H,:00E5   Lowest destination address
C539 11E600          LXI   D,:00E6   Lowest source address
C53C 1A      LC105   LDAX  D         Get byte
C53D 77              MOV   M,A       and move it
C53E 13              INX   D
C53F 23              INX   H
C540 05              DCR   B
C541 C23CC5          JNZ   :C53C     Next byte if not ready
C544 362E            MVI   M,:2E     Insert '.' after string
C546 E1              POP   H
C547 D1              POP   D
C548 C1              POP   B
C549 F1              POP   PSW
C54A C9              RET
                *
                * CALCULATE NUMBER OF DIGITS FOR OUTPUT:
                *
                * Entry: Total nr of string digits in A and C.
                *        B: Flag for INT (0) or FPT (1).
                *        Digits in 00E4 to 00E4 + A.
                * Exit:  A:  Nr of bytes for output:
                *                 INT: excl. trailing '.0'
                *                 FPT: incl. trailing '.0'
                *        HL: If last non-zero byte is not '.':
                *            points after last byte.
                *            Else: INT: points to '.'
                *                  FPT: after '.0'
                *
C54B C5      LC106   PUSH  B
C54C D5              PUSH  D
C54D 21E400          LXI   H,:00E4   Startaddr string
C550 5F              MOV   E,A       Total nr of digits in E
C551 1600            MVI   D,:00
C553 19              DAD   D         Calc end of string
C554 7E      LC107   MOV   A,M       Get digit
C555 FE30            CPI   :30
```

```
C557 C25FC5             JNZ   :C55F     If non-zero
C55A 2B                 DCX   H         Points to previous digit
C55B 0D                 DCR   C         Decr nr of digits
C55C C354C5             JMP   :C554     Again till non-zero found

                * If non-zero digit found:

C55F FE2E       LC108   CPI   :2E       '.' ?
C561 23                 INX   H
C562 C26FC5             JNZ   :C56F     Abort if not
C565 2B                 DCX   H         Pnts after last non-zero,
                                        non-'.' digit
C566 0D                 DCR   C         Excl. '.'
C567 05                 DCR   B
C568 C26FC5             JNZ   :C56F     If INT case
C56B 23                 INX   H         ) If FPT case: pnts
C56C 23                 INX   H         ) after '.0'
C56D 0C                 INR   C
C56E 0C                 INR   C         Incl. '.0'
C56F 79         LC109   MOV   A,C       Nr of digits for output
C570 D1                 POP   D
C571 C1                 POP   B
C572 C9                 RET
                *
                ********************************
                * INPUT INTEGER NUMBER TO MACC *
                ********************************
                *
                * Read string of digits from line and convert
                * it to binary in MACC.
                *
                * Entry: BC points to input character.
                * Exit:  BC points after INT number.
                *        ADEHL preserved.
                *        CY=1: there were digits.
                *        CY=0: No digits.
                *
C573 37         ICB     STC
C574 F5                 PUSH  PSW
C575 D5                 PUSH  D
C576 E5                 PUSH  H
C577 CD98C5             CALL  :C598     Clear MACC and 00E3-E6.
C57A CD73C0     LC110   CALL  :C073     Get digit from line
C57D D630               SUI   :30       Convert ASCII to binary
C57F DA90C5             JC    :C590     )
C582 FE0A               CPI   :0A       ) Abort if no number
C584 D290C5             JNC   :C590     )
C587 2110C6             LXI   H,:C610   Addr INT(10)
C58A CDA5C5             CALL  :C5A5     MACC=MACC*10 + digit
C58D C37AC5             JMP   :C57A     Next digit
C590 15         LC111   DCR   D
C591 14                 INR   D
C592 C2A2C2             JNZ   :C2A2     If digits: Pop, ret
C595 C3A9C2             JMP   :C2A9     If no digits: CY=0, Pop, ret
                *
                * CLEAR MACC AND 00E3-00E6:
                *
                * Both MACC and registers 00E3-E6 are loaded
                * with the value of FPT (0).
                *
                * Exit: ABCE preserved. D=0.
                *
```

```
374 C598 215EC4     LC112   LXI   H,:C45E   Addr. FPT(0)
375 C59B E7                 RST   4         Copy FPT(0) to MACC
376 C59C 0C                 DATA  :0C
377 C59D 21E300             LXI   H,:00E3
378 C5A0 E7                 RST   4         Copy FPT(0) to 00E3-E6
379 C5A1 0F                 DATA  :0F
380 C5A2 1600               MVI   D,:00     Clear D
381 C5A4 C9                 RET
382                 *
383                 * MACC = MACC*10 + DIGIT FROM LINE:
384                 *
385                 * Entry: HL: points to INT (10).
386                 *         A:  digit to be added.
387                 *
388 C5A5 E7         LC113   RST   4         MACC=MACC*10 (INT)
389 C5A6 54                 DATA  :54
390 C5A7 0C         LC114   INR   C
391 C5A8 15                 DCR   D
392 C5A9 32E600             STA   :00E6     Digit in lobyte E3-E6
393 C5AC 21E300             LXI   H,:00E3
394 C5AF E7                 RST   4         Add (E3-E6) to MACC (INT)
395 C5B0 4E                 DATA  :4E
396 C5B1 C9                 RET
397                 *
398                 ****************************************
399                 * CONVERT INTEGER NUMBER FOR OUTPUT *
400                 ****************************************
401                 *
402                 * Places ASCII string from INT MACC contents in
403                 * output buffer 00E4-F0.
404                 * 00E4 is sign, 00E5 is '.', 00E6-F0 is value,
405                 * 00F1 is nr of digits.
406                 *
407                 * Exit: A: Number of digits.
408                 *       BCDEHL preserved.
409                 *
410 C5B2 C5         IBC     PUSH  B
411 C5B3 D5                 PUSH  D
412 C5B4 E5                 PUSH  H
413 C5B5 CD1EC2             CALL  :C21E     Save MACC to TOS
414 C5B8 CDE0C5             CALL  :C5E0     Abs.value of MACC in reg
415                                          A,B,C,D; Prepare 00E4-E6
416 C5BB CD1EC2     LC115   CALL  :C21E     Save MACC to TOS
417 C5BE 2110C6             LXI   H,:C610   Addr INT(10)
418 C5C1 E7                 RST   4         MACC = remainder MACC/10
419 C5C2 5A                 DATA  :5A
420 C5C3 E7                 RST   4         Copy MACC to reg A,B,C,D
421 C5C4 15                 DATA  :15
422 C5C5 7A                 MOV   A,D       Lobyte in A
423 C5C6 CDFAC5             CALL  :C5FA     Digit into 00E5-F0
424 C5C9 CD34C2             CALL  :C234     Retrieve MACC from TOS
425 C5CC E7                 RST   4         MACC = MACC/10 (INT)
426 C5CD 57                 DATA  :57
427 C5CE E7                 RST   4         Copy MACC to reg A,B,C,D
428 C5CF 15                 DATA  :15
429 C5D0 B0                 ORA   B
430 C5D1 B1                 ORA   C
431 C5D2 B2                 ORA   D
432 C5D3 C2BBC5             JNZ   :C5BB     Again if <>0
433 C5D6 CD06C6             CALL  :C606     '.' in 00E5; length in 00F1
434 C5D9 CD34C2             CALL  :C234     Retrieve MACC from TOS
435 C5DC E1                 POP   H
```

```
C5DD D1                 POP   D
C5DE C1                 POP   B
C5DF C9                 RET
                *
                * PREPARE 00E4-E6:
                *
                * 00E4-E6 is set to +00 or -00, depending on sign
                * of contents MACC. In the MACC remains the absolute
                * value. The registers A,B,C,D contain the original
                * contents of the MACC.
                *
                * Exit: E=0. HL preserved. AFBCD corrupted.
                *
C5E0 E5         LC116   PUSH  H
C5E1 21E400             LXI   H,:00E4
C5E4 E7                 RST   4           Copy MACC to reg A,B,C,D
C5E5 15                 DATA  :15
C5E6 B7                 ORA   A           Set flags on sign
C5E7 362B               MVI   M,:2B       '+' in 00E4
C5E9 F2F0C5             JP    :C5F0       Jump if nr is positive
C5EC 362D               MVI   M,:2D       Else '-' in 00E4
C5EE E7                 RST   4           and make contents MACC pos.
C5EF 60                 DATA  :60
C5F0 23         LC117   INX   H
C5F1 3630               MVI   M,:30       0 in 00E5
C5F3 23                 INX   H
C5F4 3630               MVI   M,:30       0 in 00E6
C5F6 1E00               MVI   E,:00       Digit count is 0
C5F8 E1                 POP   H
C5F9 C9                 RET
                *
                * STORE DIGIT IN OUTPUT BUFFER 00E5-00F0:
                *
                * Entry: Digit in A.
                * Exit:  Digit in 00E5-F0 as most sign. digit.
                *        E:  Count of digit in buffer.
                *        BCDHL preserved. AF corrupted.
                *
C5FA E5         LC118   PUSH  H
C5FB F5                 PUSH  PSW
C5FC CD1AC5             CALL  :C51A       Move contents buffer right
C5FF F1                 POP   PSW
C600 C630               ADI   :30         Make digit ASCII
C602 77                 MOV   M,A         Digit in 00E5 inserted.
C603 1C                 INR   E           Update digit count.
C604 E1                 POP   H
C605 C9                 RET
                *
                * ADD A '.' TO A DIGIT STRING IN OUTPUTBUFFER:
                *
                * A '.' is placed at the beginning of a digit
                * string in the output buffer. The length of the
                * string is stored in 00F1.
                *
                * Entry: E:  Digit count.
                *        HL: Points to 00E5.
                * Exit:  A:  Count.
                *        BCDEHL preserved.
                *
C606 CD1AC5     LC119   CALL  :C51A       Move contents outbuf right
                                          one position
C609 362E               MVI   M,:2E       '.' at begin of string
```

```
498 C60B 7B                     MOV   A,E       Get digit count
499 C60C 32F100                 STA   :00F1     Store it in buffer
500 C60F C9                     RET
501                   *
502                   * DATA:
503                   *
504 C610 00           I10       DATA  :00       INT (10)
505 C611 00                     DATA  :00
506 C612 00                     DATA  :00
507 C613 0A                     DATA  :0A
508                   *
509                   *
510                   *
511 C614                        END
```

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

FP0	C45E	FP1	C462	FP2	C466	FP3	C46A
FP4	C46E	FP5	C472	FP6	C476	FP7	C47A
FP8	C47E	FP9	C482	I10	C610	IBC	C5B2
ICB	C573	LC100	C502	LC101	C513	LC102	C51A
LC103	C525	LC104	C531	LC105	C53C	LC106	C54B
LC107	C554	LC108	C55F	LC109	C56F	LC110	C57A
LC111	C590	LC112	C598	LC113	C5A5	LC114	C5A7
LC115	C5BB	LC116	C5E0	LC117	C5F0	LC118	C5FA
LC119	C606	LC234	C437	LC238	C45A	LC91	C498
LC92	C4A3	LC93	C4AF	LC94	C4C3	LC95	C4C7
LC96	C4D5	LC97	C4D8	LC98	C4E1	LC99	C4FF
PRTY	C486						

```
002                             ORG   :C614
003                     *
004                     *
005                     *
006                     ******************************
007                     * INPUT HEX NUMBER TO MACC *
008                     ******************************
009                     *
010                     * Reads a sequence of hex digits and converts
011                     * them into MACC.
012                     *
013                     * Entry: C points to input.
014                     * Exit:  CY=1: There was a digit.
015                     *        CY=0: No digit.
016                     *        C points to next input.
017                     *        ABDEHL preserved.
018                     *
019 C614 37             HCB     STC
020 C615 F5                     PUSH  PSW
021 C616 D5                     PUSH  D
022 C617 E5                     PUSH  H
023 C618 CD98C5                 CALL  :C598     Clear MACC and 00E3-E6
024 C61B CD73C0         LC120   CALL  :C073     Get digit from line
025 C61E D630                   SUI   :30       )
026 C620 DA90C5                 JC    :C590     )
027 C623 FE0A                   CPI   :0A       ) Check if hex number
028 C625 DA34C6                 JC    :C634     )
029 C628 D607                   SUI   :07       ) Abort via C590 if not
030 C62A FE0A                   CPI   :0A       )
031 C62C DA90C5                 JC    :C590     )
032 C62F FE10                   CPI   :10       )
033 C631 D290C5                 JNC   :C590     )
034 C634 213DC6         LC121   LXI   H,:C63D   Addr INT(4)
035 C637 CD41C6                 CALL  :C641     Insert digit at low end
036                                             MACC
037 C63A C31BC6                 JMP   :C61B     Get next digit
038                     *
039                     * DATA:
040                     *
041 C63D 00             I4      DATA  :00       INT (4)
042 C63E 00                     DATA  :00
043 C63F 00                     DATA  :00
044 C640 04                     DATA  :04
045                     *
046                     * ENTER HEX DIGIT AT LOW END MACC:
047                     *
048                     * Entry: HL points to a 4-byte number.
049                     *        A  contains a digit.
050                     * Exit:  HL = 00E3.
051                     *        C is incremented, D decremented.
052                     *        ABE preserved.
053                     *
054 C641 F5             LC122   PUSH  PSW
055 C642 C5                     PUSH  B
056 C643 D5                     PUSH  D
057 C644 E7                     RST   4         Copy MACC to reg A,B,C,D
058 C645 15                     DATA  :15
059 C646 E6F0                   ANI   :F0       Check if value too high
060 C648 C44BC0                 CNZ   :C04B     Then overflow error
061 C64B D1                     POP   D
062 C64C C1                     POP   B
063 C64D F1                     POP   PSW
```

```
064 C64E E7                     RST   4          Shift left
065 C64F 6F                     DATA  :6F
066 C650 C3A7C5                 JMP   :C5A7      Add digit to MACC
067                     *
068                     ****************************************
069                     * CONVERT HEX MACC TO ASCII FOR OUTPUT *
070                     ****************************************
071                     *
072                     * Converts a HEX number in MACC into its ASCII
073                     * representation into the output buffer.
074                     * Not significant leading zeroes are cancelled.
075                     *
076                     * Exit: BCDEHL preserved. AF corrupted.
077                     *       Length output string in 00E3.
078                     *       Output string in 00E4-00EB.
079                     *
080 C653 C5             HBC     PUSH  B
081 C654 D5                     PUSH  D
082 C655 E5                     PUSH  H
083 C656 CD8DC6                 CALL  :C68D      Get startaddr DECBUF in HL
084 C659 E7                     RST   4          Copy MACC to reg A,B,C,D
085 C65A 15                     DATA  :15
086 C65B CD6AC6                 CALL  :C66A      Convert A,B to ASCII
087                                              into DECBUF
088 C65E 79                     MOV   A,C
089 C65F 42                     MOV   B,D
090 C660 CD6AC6                 CALL  :C66A      Idem for C,D
091 C663 CD91C6                 CALL  :C691      Get string length in 00E3
092 C666 E1                     POP   H
093 C667 D1                     POP   D
094 C668 C1                     POP   B
095 C669 C9                     RET
096
097                     * Convert 2 hex digits:
098
099 C66A CD6EC6         LC123   CALL  :C66E      Convert 1st digit
100 C66D 78                     MOV   A,B        Get 2nd one
101 C66E F5             LC124   PUSH  PSW
102 C66F 1F                     RAR
103 C670 1F                     RAR
104 C671 1F                     RAR
105 C672 1F                     RAR              Shift high nibble in low
106 C673 CD77C6                 CALL  :C677      Convert it to ASCII
107 C676 F1                     POP   PSW        Restore both nibbles
108 C677 E60F           LC125   ANI   :0F        Low nibble only
109 C679 FE0A                   CPI   :0A
110 C67B DA80C6                 JC    :C680      If 0 < digit < 9
111 C67E C607                   ADI   :07        Add 7 for A < digit < F
112 C680 C630           LC126   ADI   :30        Convert to ASCII
113 C682 77                     MOV   M,A        Into DECBUF
114 C683 23                     INX   H          Incr pointer
115 C684 FE30                   CPI   :30
116 C686 C0                     RNZ              Abort if digit <> 0
117
118                     * If 1st digit is zero:
119
120 C687 7D                     MOV   A,L        Get lsbyte buffer pointer
121 C688 FEE5                   CPI   :E5        1st digit in buffer?
122 C68A C0                     RNZ              Abort if not
123 C68B 2B                     DCX   H          Else: cancel non-sign. 0's
124 C68C C9                     RET
125
```

```
                        * Get startaddress output buffer:
                        *
C68D 21E400     LC127   LXI   H,:00E4   Startaddr in HL
C690 C9                 RET
                        *
                        * CALCULATE LENGTH OF STRING IN OUTPUT BUFFER:
                        *
                        * Entry: L: lobyte of address last digit in buffer.
                        * Exit:  BCDEHL preserved. AF corrupted.
                        *        Length is stored in 00E3.
                        *
C691 7D         LC128   MOV   A,L       Get lobyte addr last digit
C692 D6E4               SUI   :E4       Minus beginaddr
C694 FE01               CPI   :01
C696 CE00               ACI   :00       Length min. 1
C698 32E300             STA   :00E3     Store length in DECBUF
C69B C9                 RET
                        *
                        ******************************************
                        * RESTORE A, ADD '.' AFTER DIGIT STRING *
                        ******************************************
                        *
                        * Part of PRTY (C486).
                        *
C69C 90         LC129   SUB   B         Restore A
C69D C331C5             JMP   :C531     Move string left 1 pos,
                                        insert '.'
                        *
                        ****************************************
                        * PRINT CHARACTER, INPUT A TEXT LINE *
                        ****************************************
                        *
                        * Part of Run 'INPUT' (0E3D6).
                        *
                        * Entry: Character in A.
                        * Exit:  BC preserved.
                        *
C6A0 C5         PINPLN  PUSH  B
C6A1 CD1FDD             CALL  :DD1F     Print char; input textline
C6A4 C1                 POP   B
C6A5 C9                 RET
                        *
C6A6 FF                 DATA  :FF
C6A7 FF                 DATA  :FF
                        *
                        **********************
                        * DATA FOR 'RANDOM' *
                        **********************
                        *
C6A8 00         RNDA    DATA  :00       Random number constant A
C6A9 00                 DATA  :00
C6AA 00                 DATA  :00
C6AB 3B                 DATA  :3B
                        *
C6AC 07         RNDB    DATA  :07       Random number constant B
C6AD 73                 DATA  :73
C6AE 59                 DATA  :59
C6AF 41                 DATA  :41
                        *
C6B0 01         IROR    DATA  :01       OR mask  (FPT (1))
C6B1 80                 DATA  :80
C6B2 00                 DATA  :00
```

```
188 C6B3 00                 DATA  :00
189                 *
190                 ******************************
191                 * part of READ BLOCK (D340) *
192                 ******************************
193                 *
194                 * Exit if no loading errors.
195                 *
196 C6B4 E3         MPT26   XTHL
197 C6B5 37                 STC             CY=1: no error
198 C6B6 E1         LBK30   POP   H
199 C6B7 D1                 POP   D
200 C6B8 C1                 POP   B
201 C6B9 C9                 RET
202                 *
203                 ******************
204                 * part of 2E8DE *
205                 ******************
206                 *
207 C6BA CD91CE     SPT02   CALL  :CE91     Go and set screen bits for
208                                         mode 1
209 C6BD C338E1             JMP   :E138     (2) Pop all, ret.
210                 *
211                 *
212                 *  ================
213                 *** BANK SWITCHING ***
214                 *  ================
215                 *
216                 *
217                 ***************************
218                 * MATH. RESTART (RST 4) *
219                 ***************************
220                 *
221                 * This, and the following routines, switch the
222                 * paged banks of ROM. They are entered via
223                 * RST x; DATA xx instructions.
224                 *
225 C6C0 E1         MARST   POP   H
226 C6C1 F3                 DI
227 C6C2 224300             SHLD  :0043     Save HL
228 C6C5 F5                 PUSH  PSW
229 C6C6 E1                 POP   H
230 C6C7 224100             SHLD  :0041     Save PSW
231 C6CA 2640               MVI   H,:40     ROM bank 1 select bits
232 C6CC 3AD400             LDA   :00D4     Offset of start HW/SW vector
233                 *
234                 * ROM BANK SWITCHING:
235                 *
236                 * This routine is generally used by all Restarts
237                 * using ROM bank switching.
238                 *
239 C6CF E3         MRS10   XTHL
240 C6D0 86                 ADD   M         Add entry number
241 C6D1 23                 INX   H
242 C6D2 E3                 XTHL
243 C6D3 6F                 MOV   L,A       Complete entry point address
244 C6D4 3A4000             LDA   :0040     Old bank select port status
245 C6D7 F5                 PUSH  PSW       Save it
246 C6D8 E63F               ANI   :3F       Keep other bits
247 C6DA B4                 ORA   H         Add new select bits
248 C6DB 324000             STA   :0040     Update memory
249 C6DE 3206FD             STA   :FD06     Update port
```