```
ØØØØ  F3          X0000  DI
ØØØ1  C3 9Ø 27    X0001  JMP   X2790
ØØØ4  11 1C 2Ø    X0004  LXI   D,X2Ø1C
ØØØ7  1C                 INR   E
RST 1 ØØØ8  7E           MOV   A,M   PUT NXT CHAR FROM STMT AREA INTO ACC
ØØØ9  E3                 XTHL  SAVE PTR ON STACK, FETCH RETURN ADDR FROM STACK
ØØØA  BE                 CMP   M COMPARE WITH CHAR AT ADDR AFTER RST 1 SUBR CALL
ØØØB  23          X0008  INX   H ADVANCE HL TO RETURN TO ADDR AFTER ARGUMENT BYTE
ØØØC  E3                 XTHL  SWAP PTR'S AGAIN
ØØØD  C2 A9 Ø4           JNZ   X04A9 SYNTAX ERROR MESSAGE
RST 2 ØØ1Ø  23           INX   H nxt char.  RZ null or colon  } skip spaces
ØØ11  7E                 MOV   A,M          RC DIGIT
ØØ12  FE 3A              CPI   A':'         RNC NOT A DIGIT
ØØ14  DØ                 RNC   RETURN NO CARRY, CHAR POINTED BY H,L IS ':' or larger char
ØØ15  C3 Ø6 Ø7           JMP   X0706
RST 3 ØØ18  F5    X0018  PUSH  PSW   PUT CHAR TO PRINT ON STACK
ØØ19  3A A2 Ø3           LDA   X03A2 } LOAD ACC FROM PRINT SUPPRESSION FLAG + TEST FLAG
ØØ1C  B7                 ORA   A
ØØ1D  C3 CB Ø6           JMP   X06CB JMP TO OUTPUT CONSOLE BYTE-TRANSFER ROUTINE
RST 4 ØØ2Ø  7C           MOV   A,H   COMPARE HL + DE
ØØ21  92                 SUB   D
ØØ22  CØ                 RNZ         RZ HL = DE
ØØ23  7D                 MOV   A,L   RC HL < DE
ØØ24  93                 SUB   E     RNC HL ≥ DE
ØØ25  C9                 RET   (# of nulls / char count for output ctrl'd by TAB(I))
      ØØ26  Ø1 ØØ 3A X0026 LXI   B,X3A00  DBL-PREC STARTS AT ØY0E, SNG-PREC STARTS AT Ø412
RST 5 ØØ29  15           DCR   D     FETCH EXPONENT BYTE OF # IN HOLDING AREA (LOC Ø415)
ØØ2A  Ø4                 INR   B
ØØ2B  B7                 ORA   A     TEST BYTE
ØØ2C  C2 EB 1A           JNZ   X1AEB
ØØ2F  C9                 RET   RZ: EXPONENT = ØØ  MAGNITUDE = Ø
RST 6 ØØ3Ø  3A A4 Ø3     LDA   X03A4 FETCH TYPE CODE FROM FLAG AREA (INT=Ø2, SNG=Ø4, DBL=Ø8, $=Ø3, CHAR=Ø3)
ØØ33  FE Ø8              CPI   H'Ø8' COMPARE TYPE CODE WITH DBL-PRECISION TYPE CODE
ØØ35  C3 AE ØE           JMP   X0EAE CONTINUE THIS ROUTINE ELSEWHERE  RPo-SNG
RST 7 ØØ38  C9           RET   USER FUNCTION AREA
ØØ39  ØØ                 NOP   SUBSTITUTE JMP INSTRUCTION TO USER PROGRAM
ØØ3A  ØØ                 NOP
      ØØ3B  1B    X003B  √DCX  D     SGN  BA  1B1B
      ØØ3C  1B            DCX   D
      ØØ3D  BA            CMP   D     INT  BB  1CBA
      ØØ3E  1C            INR   E
      ØØ3F  Ø8            √DATA H'Ø8'
      ØØ4Ø  1B            DCX   D     ABS  BC  1BØ8
      ØØ41  EE Ø8         √XRI  H'Ø8' USR  BD  Ø8EE
      ØØ43  BØ            ORA   B
      ØØ44  1Ø            DATA  H'1Ø' FRE  BE  1ØBØ
      ØØ45  3Ø            √DATA A'Ø'
      ØØ46  14            INR   D     INP  BF  143Ø
      ØØ47  D1            POP   D
      ØØ48  1Ø            /DATA H'1Ø' POS  CØ  1ØD1
      ØØ49  AC            √XRA  H
      ØØ4A  25            DCR   H     SQR  C1  25AC
      ØØ4B  88            ADC   B     RND  C2  2688
      ØØ4C  26 98         √MVI  H,H'98'
      ØØ4E  19            DAD   D     LOG  C3  1998
      ØØ4F  F7            √RST  6
      ØØ5Ø  25            DCR   H     EXP  C4  25F7
      ØØ51  BE    X0051  √CMP  H     COS  C5  26BE
      ØØ52  26 C4         √MVI  H,H'C4' SIN C6 26C4
```

FORMAT	PURPOSE
RST 1	TEST BYTE POINTED TO
ARG BYTE	BY HL FOR EQUALITY
	WITH BYTE AFTER RST 1
	RETURN WITH HL POINTING TO NXT
	NON-SPACE BYTE

RST5 FLOATING POINT SIGN TEST
IF EXPONENT = Ø RZ
IF MSB OF MOST SIGNIFICANT FRACTION
BYTE IS ON RETURN WITH -1 IN ACC
AND SIGN BIT SET
IF MSB ' ' IS OFF SET ACC TO +1
M SIGN BIT OFF

TEST FOR VARIABLE OF
TYPE (STORED AT Ø3
-DBL
RZ - CHAR RNC - DBL RP - SNG RM - I

```
                C4                  √SIN  C6        26C4
0054  28 32                        √TAN  C7        2721
0056  27                          √ATN  C8        2736
0057  3E 27         X0057         √PEEK C9        154B
0059  4B                          √CINT CA        1C11
005A  15
005B  11 1C 45                    √CSGN CB        1C45
005E  1C
005F  6F                          √CDBL CC        1C6F
0060  1C                          √LEN  CD        13A4
0061  A4                          √STR$ CE        11CD
0062  13
0063  CD 11 8E                    √VAL  CF        148E
0066  14
0067  B0                          √ASC  D0        13B0
0068  13
0069  BE                          √CHR$ D1        13BE
006A  13
006B  CE 13                       √LEFT$ D2       13CE
006D  FF                          √RIGHT$ D3      13FF
006E  13
006F  09                          √MID$ D4        1409
0070  14
0071  79            X0071       +
0072  79                        -
0073  7C                        *
0074  7C                        /
0075  7F                        ↑
0076  50                        AND
0077  46                        OR
0078  7A                        MOD
0079  7B            X0079
007A  45            X007A       END             80
007B  4E
007C  C4 46 4F                  FOR             81
007F  D2 4E 45                  NEXT
0082  58                                        82
0083  D4 44 41                  DATA
0086  54
0087  C1                                        83
0088  49                        INPUT
0089  4E
008A  50
008B  55                                        84
008C  D4 44 49                  DIM             85
008F  CC 52 45                  READ
0092  41                                        86
0093  C4 4C 45                  LET             87
0096  D4 47 4F                  GO TO
0099  54                                        88
009A  CF
0090  52                        RUN             89
009C  55
009D  CE 49                     IF              8A
009F  C6 52                     RESTORE         8B
00A1  45
00A2  53
00A3  54
00A4  4F
00A5  52
00A6  C5
```

RELATED TO THE ARITHMETIC OPERATORS

ARITHMETIC OPERATOR PRECEDENCE TABLE

LAST LETTER OF EACH WORD HAS MSB set (1)

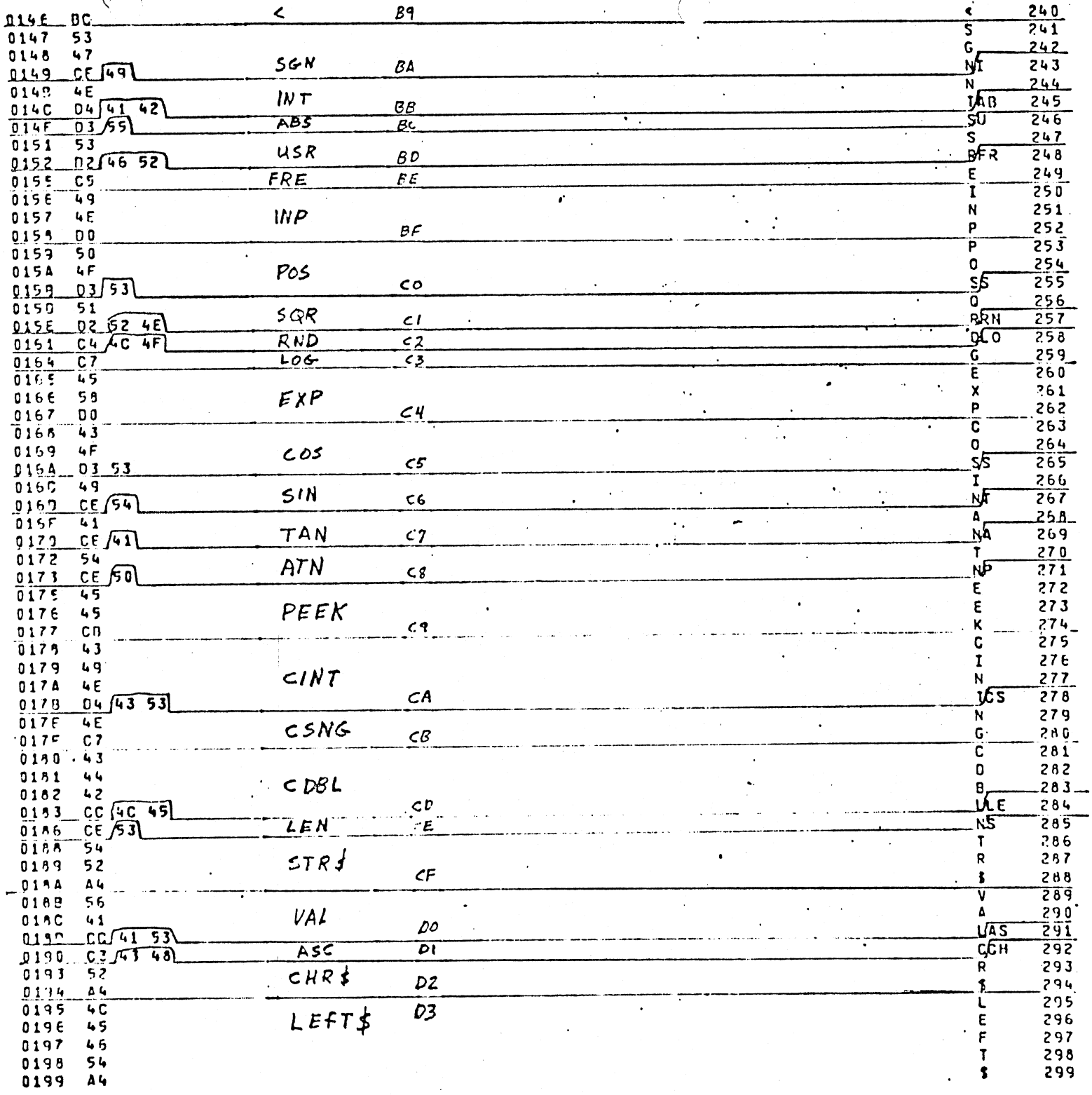
```
                                  $I      60
                                  '       61
                                  6'      62
                                  K       63
                                          64
                                    E     65
                                          66
                                          67
                                          68
                                  $       69
                                          70
                                  M       71
                                          72
                                  0       73
                                          74
                                  >       75
                                          76
                                  N       77
                                          78
                                          79
                                          80
                                          81
                                          82
                                          83
                                          84
                                          85
                                          86
                                  P       87
                                  F       88
                                          89
                                          90
                                          91
                                          92
                                          93
                                          94
                                          95
                                          96
                                          97
                                          98
                                          99
                                         100
                                         101
                                         102
                                         103
                                         104
                                         105
                                         106
                                         107
                                         108
                                         109
                                         110
                                         111
                                         112
                                         113
                                         114
                                         115
                                         116
                                         117
                                         118
                                         119
```

```
00A7  47
00A8  4F
00A9  53
00AA  55
00AB  C2 [52 45]   GOSUB   8C
00AF  54
00AF  55
00B0  52
00B1  CE [52]      RETURN  8D
00B3  45
00B4  CD [53 54]   REM     8E
00B7  4F
00B8  D0           STOP    8F
00B9  45
00BA  4C
00BB  53
00BC  C5           ELSE    90
00BD  54
00BE  52
00BF  4F
00C0  CE [54]      TRON    91
00C2  52
00C3  4F
00C4  46
00C5  C6 [53]      TROFF   92
00C7  57
00C8  41
00C9  D0           SWAP    93
00CA  45
00CB  52
00CC  41
00CD  53
00CE  C5           ERASE   94
00CF  44
00D0  45
00D1  4E
00D2  53
00D3  54
00D4  D2 [44 45]   DEFSTR  95
00D7  46
00D8  49
00D9  4E
00DA  D4 [44 45]   DEFINT  96
00DD  46
00DE  53
00DF  4E
00E0  C7           DEFSNG  97
00E1  44
00E2  45
00E3  4E
00E4  44
00E5  42
00E6  CC [45 44]   DEFDBL  98
00E9  49           EDIT    99
00EA  D4 [4F 55]   OUT     9A
00ED  D4 [4F CE]   ON      9B
00F0  4E
00F1  55
00F2  4C
00F3  CC [57 41]   NULL    9C
```

```
G   120
O   121
S   122
U   123
ORE 124
T   125
U   126
R   127
NR  128
E   129
MST 130
C   131
P   132
E   133
L   134
S   135
E   136
T   137
R   138
O   139
NT  140
R   141
C   142
F   143
FS  144
H   145
A   146
P   147
E   148
R   149
A   150
S   151
E   152
C   153
E   154
F   155
S   156
T   157
RDE 158
F   159
I   160
N   161
TDE 162
F   163
S   164
N   165
G   166
D   167
E   168
F   169
D   170
B   171
LED 172
I   173
TOU 174
TON 175
N   176
U   177
L   178
LWA 179
```

```
00F6  49                WAIT   9D
00F7  D4 44 45          DEF    9E
00FA  CF 50
00FC  4F
00FD  4B                POKE   9F
00FE  C5
00FF  50          X00FF
0100  52
0101  49                PRINT  A0
0102  4E
0103  D4 43 4F          CONT   A1
0106  4E
0107  D4 4C 49
010A  53                LIST   A2
010B  D4 53 54
010E  4C
010F  45
0110  54                DELETE A3
0111  C5
0112  43
0113  4C
0114  45
0115  41                CLEAR  A4
0116  D2 4E 45          NEW    A5
0119  D7
011A  54
011B  41
011C  42                TABC   A6
011D  A8
011E  54          X011E
011F  CF                TO     A7
0120  46
0121  CE 53             FN     A8
0123  50
0124  43
0125  A8                SPC(   A9
0126  55
0127  53
0128  49
0129  4E                USING  AA
012A  C7
012B  54
012C  48
012D  45                THEN   AB
012E  CE 4F
0130  4F                NOT    AC
0131  D4 45 50          STEP   AD
0134  45
0135  00                +      AE
0136  A9                -      AF
0137  AC                *      B0
0138  AA                /      B1
0139  AF                ↑      B2
013A  DE 41             AND    B3
013C  4E
013D  C4 4F 52          OR     B4
0140  4D                MOD    B5
0141  4F
0142  C4 0C BE          >      B6 B7
0145  BD                <      B8
```

```
I    180
NDE  181
FOR  182
O    183
K    184
E    185
P    186
R    187
I    188
N    189
TCO  190
N    191
ULI  192
S    193
TDE  194
L    195
E    196
T    197
E    198
C    199
L    200
E    201
A    202
RNE  203
W    204
T    205
A    206
B    207
(    208
T    209
O    210
F    211
NS   212
P    213
C    214
(    215
U    216
S    217
I    218
N    219
G    220
T    221
H    222
E    223
NXT  224
O    225
TST  226
E    227
P    228
+    229
-    230
*    231
/    232
↑A   233
N    234
DDR  235
H    236
O    237
D(>  238
=    239
```


014E BC
0147 53
0148 47
0149 CE 47
014A 4E
014C D4 41 42
014F D3 53
0151 53
0152 D2 46 52
0155 C5
0156 49
0157 4E
0159 D0
0159 50
015A 4F
0159 D3 53
0150 51
015E D2 52 4E
0161 C4 4C 4F
0164 C7
0165 45
016E 58
0167 D0
0168 43
0169 4F
016A D3 53
016C 49
016D CE 54
016F 41
0170 CE 41
0172 54
0173 CE 50
0175 45
0176 45
0177 CD
0178 43
0179 49
017A 4E
017B D4 43 53
017E 4E
017F C7
0180 43
0181 44
0182 42
0183 CC 4C 45
0186 CE 53
0188 54
0189 52
018A A4
018B 56
018C 41
018D CC 41 53
0190 C3 48 52
0193 52
0194 A4
0195 4C
0196 45
0197 46
0198 54
0199 A4
SGN
INT
ABS
USR
FRE
INP
POS
SQR
RND
LOG
EXP
COS
SIN
TAN
ATN
PEEK
CINT
CSNG
CDBL
LEN
STR$
VAL
ASC
CHR$
LEFT$
BA
BB
BC
BD
BE
BF
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
CA
CB
CD
E
CF
D0
D1
D2
D3
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299

019A	52		RIGHT$ D4			R	300
019B	49					I	301
019C	47					G	302
019D	48					H	303
019E	54					T	304
019F	A4					$	305
01A0	4D		MID$ D5			M	306
01A1	49					I	307
01A2	44					D	308
01A3	A4					$	309
01A4	00					NULL	310
01A5	F7	X01A5					311
01A6	07		√ END	80	Ø7F7		312
01A7	07	ADDR OF SUBROUTINE	√ FOR	81	Ø7Ø2		313
01A8	07					H	314
01A9	D7		√ NEXT	82	ØBD7		315
01AA	0B						316
01AB	8C		√ DATA	83	Ø98C		317
01AC	09						318
01AD	0F		INPUT	84	ØBØF		319
01AE	0B						320
01AF	0B 0E		√ DIM	85	ØEDB	(	321
0101	3E 0B		READ	86	ØB36	6	322
0103	AB		LET	87	Ø9AB	+	323
0104	09						324
0105	5A		√ GOTO	88	Ø95A	Z	325
0106	09						326
0107	30		√ RUN	89	Ø93D	=	327
0108	09						328
0109	16 0A		√ IF	8A	ØA16		329
010B	E1		√ RESTORE	8B	Ø7E1		330
010C	07						331
010D	49		√ GOSUB	8C	Ø949	I	332
010E	09						333
010F	76		√ RETURN	8D	Ø976		334
01C0	09						335
01C1	8E		√ REM	8E	Ø98E		336
01C2	09						337
01C3	F5		√ STOP	8F	Ø7F5		338
01C4	07						339
01C5	8E		√ ELSE	90	Ø98E		340
01C6	09						341
01C7	3E 0B		√ TRON	91	ØB3E	>	342
01C9	3F		√ TROFF	92	ØB3F	?	343
01CA	0B						344
01CB	44		√ SWAP	93	Ø844	D	345
01CC	08						346
01CD	7A		√ ERASE	94	Ø87A		347
01CE	08						348
01CF	A8		√ DEF STR	95	Ø8A8	(	349
01D0	08						350
01D1	AB		√ DEF INT	96	Ø8AB	+	351
01D2	08						352
01D3	AE		√ DEF SGN	97	Ø8AE	.	353
01D4	08						354
01D5	B1		√ DEF DBL	98	Ø8B1	1	355
01D6	08						356
01D7	0C		√ EDIT	99	170C		357
01D8	17						358
01D9	3B		√ OUT	9A	143B	;	359

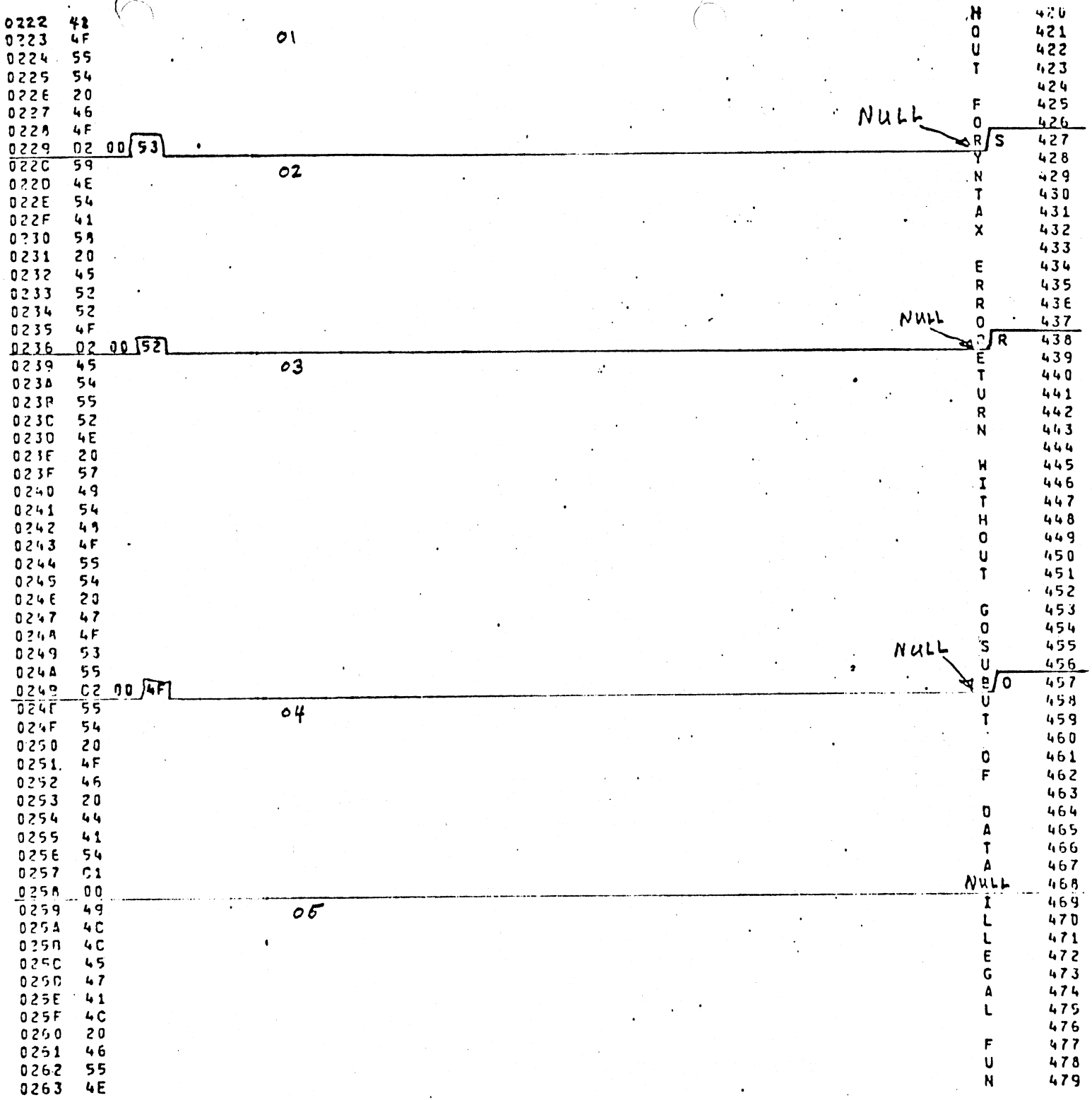
END OF TABLE

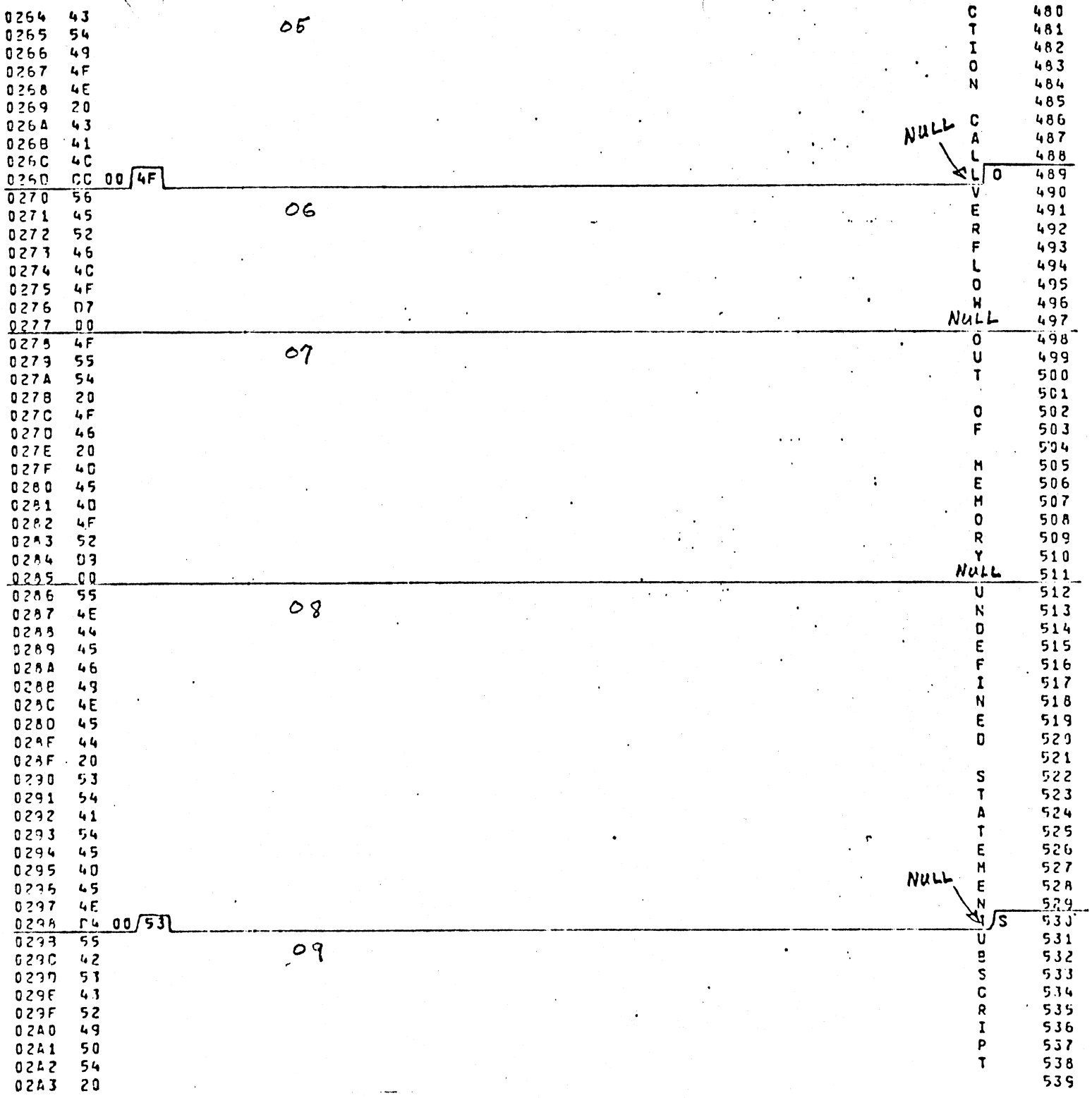
```
01DA 14                √ OUT      9A   143B                          360
01DC FA 09 (30)        √ ON       9B   09FA                     D    361
01DF 03                √ NULL     9C   0830                          362
01DF 41                                                         A    363
01E0 14                √ WAIT     9D   1441                          364
01E1 0A 10 (52)        √ DEF      9E   10DA                     Z R  365
01E4 15                √ POKE     9F   1552                          366
01E5 4C                                                         L    367
01E6 0A                √ PRINT    AD   0A4C                          368
01E7 1D                                                              369
01E8 08                √ CONT     A1   081D                          370
01E9 A8                                                         (    371
01EA 14                √ LIST     A2   14A8                          372
01EB 18                                                              373
01EC 15                √ DELETE   A3   151B                          374
01ED 14                                                              375
01EE 09                √ CLEAR    A4   0914                          376
01EF 99                                                              377
01F0 05                √ NEW      A5   0599                          378
01F1 6F   X01F1                                                      379
01F2 1C                √ TABC     A6   1C6F = CDBL                   380
01F3 00                                                              381
01F4 00                √ TO       A7   0000 - ADDR WHERE BASIC ORG'S 382
01F5 11 1C (88)        √ FN       A8   1C11 = CINT                   383
01F8 1C                √ SPCC     A9   1C88 = STRING VALUE CHECK     384
01F9 45                                                         E    385
01FA 1C                √ USING    AA   1C45 = CSGN                  386
01FB 4A   X01FB                                                 J    387
01FC 1F (43)           √ THEN     AB   1E4A = DBL-PREC ADDITION  C    388
01FE 1E (83)           √ NOT      AC   1E43 = DBL-PREC SUBTRACTION   389
0200 1F                √ STEP     AD   1F83 = DBL-PREC MULTIPLY      390
0201 C9                                                         I    391
0202 1F                √ +        AE   1FC9 = DBL-PREC DIVIDE        392
0203 0A                                                              393
0204 1C                √ -        AF   1C0A = DBL-PREC # MAGNITUDE COMPARISON (SIGN TEST 394 FOR RESULT OF SUBTRACTION)
0205 9C   X0205                                                      395
0206 18                √ *        B0   189C = SNG-PREC ADDITION      396
0207 99                                                              397
0208 19                √ /        B1   1899 = SNG-PREC SUBTRACTION   398
0209 03 19             √ ↑        B2   1903 = SNG-PREC MULTIPLY  S   399
020B 2E 1A             √ AND      B3   1A2E = SNG-PREC DIVIDE    .   400
020D A1                                                         !    401
020E 18                √ OR       B4   18A1 = SNG-PREC # MAGNITUDE COMPARISON 402 (SIGN TEST FOR RESULT OF SUBTRACTION)
020F 55   X020F                                                 U    403
0210 10                √ MOD      B5   1055 = INTEGER ADDITION       404
0211 49                                                         I    405
0212 10                √ \        B6   1049 = INTEGER SUBTRACTION    406
0213 75                                                              407
0214 10                √ >        B7   1075 = INTEGER MULTIPLY       408
0215 BC                                                         <    409
021E 00                √ =        B8   00BC = INTEGER DIVIDE         410
0217 CC 18 00          √ <        B9   18CC = INTEGER # MAGNITUDE COMPARISON  L  411 (SIGN TEST FOR RESULT OF SUBTRACTION)
021A 4E                01                                        N    412
021B 45                                                         E    413
021C 52                                                         X    414
021D 54                                                         T    415
021E 20   X021E                                                      416
021F 57                                                         W    417
0220 49                                                         I    418
0221 54                                                         T    419
```

NULL

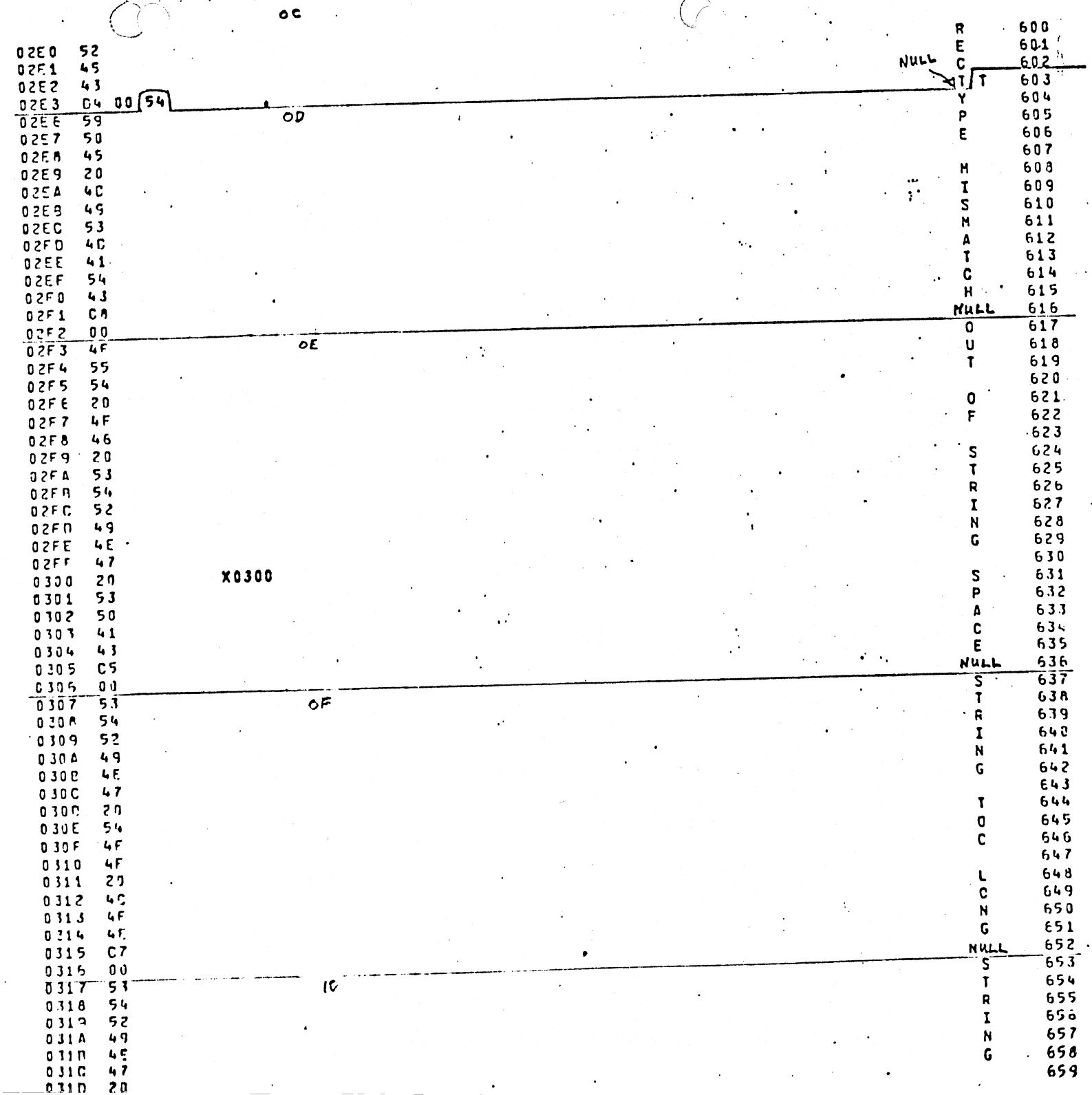
LAST CHAR IN EACH ERROR MESSAGE HAS MSB SET (1)

ERROR MESSAGES


0222 48
0223 4F
0224 55
0225 54
022E 20
0227 46
0228 4F
0229 02 00 53
022C 59
022D 4E
022E 54
022F 41
0230 58
0231 20
0232 45
0233 52
0234 52
0235 4F
0236 02 00 52
0239 45
023A 54
023B 55
023C 52
023D 4E
023E 20
023F 57
0240 49
0241 54
0242 49
0243 4F
0244 55
0245 54
024E 20
0247 47
0248 4F
0249 53
024A 55
024B 02 00 4F
024E 55
024F 54
0250 20
0251 4F
0252 46
0253 20
0254 44
0255 41
025E 54
0257 01
0258 00
0259 49
025A 4C
025B 4C
025C 45
025D 47
025E 41
025F 4C
0260 20
0261 46
0262 55
0263 4E
01
02
03
04
05
NULL
NULL
NULL
H 420
O 421
U 422
T 423
424
F 425
O 426
R S 427
Y 428
N 429
T 430
A 431
X 432
433
E 434
R 435
R 43E
O 437
R 438
E 439
T 440
U 441
R 442
N 443
444
W 445
I 446
T 447
H 448
O 449
U 450
T 451
452
G 453
O 454
S 455
U 456
B O 457
U 458
T 459
460
O 461
F 462
463
D 464
A 465
T 466
A 467
NULL 468
I 469
L 470
L 471
E 472
G 473
A 474
L 475
476
F 477
U 478
N 479


05
06
07
08
09
NULL
NULL
NULL
NULL
00 4F
00 53

02A4	4F	09		O	540
02A5	55			U	541
02A6	54			T	542
02A7	20				543
02A8	4F			O	544
02A9	46			F	545
02AA	20				546
02AB	52			R	547
02AC	41			A	548
02AD	4E			N	549
02AE	47			G	550
02AF	C5			E	551
02B0	00			NULL	552
02B1	52	0A		R	553
02B2	45			E	554
02B3	44			D	555
02B4	49			I	556
02B5	4D			M	557
02B6	45			E	558
02B7	4E			N	559
02B8	53			S	560
02B9	49			I	561
02BA	4F			O	562
02BB	4E			N	563
02BC	45			E	564
02BD	44			D	565
02BE	20				566
02BF	41			A	567
02C0	52			R	568
02C1	52			R	569
02C2	41			A	570
02C3	09			Y	571
02C4	00			NULL	572
02C5	44	0B		C	573
02C6	49			I	574
02C7	56			V	575
02C8	49			I	576
02C9	53			S	577
02CA	49			I	578
02CB	4F			O	579
02CC	4E			N	580
02CD	20				581
02CE	42			B	582
02CF	59			Y	583
02D0	20				584
02D1	5A			Z	585
02D2	45			E	586
02D3	52			R	587
02D4	CF			O	588
02D5	00			NULL	589
02D6	49	0C		I	590
02D7	4C			L	591
02D8	4C			L	592
02D9	45			E	593
02DA	47			G	594
02DB	41			A	595
02DC	4C			L	596
02DD	20				597
02DE	44			C	598
02DF	49			I	599


0C
02E0 52
02E1 45
02E2 43
02E3 04 00 54
0D
02EE 59
02E7 50
02E8 45
02E9 20
02EA 4C
02EB 49
02EC 53
02ED 4C
02EE 41
02EF 54
02F0 43
02F1 CA
02F2 00
0E
02F3 4F
02F4 55
02F5 54
02FE 20
02F7 4F
02F8 46
02F9 20
02FA 53
02FB 54
02FC 52
02FD 49
02FE 4E
02FF 47
X0300
0300 20
0301 53
0302 50
0303 41
0304 43
0305 C5
0306 00
0F
0307 53
0308 54
0309 52
030A 49
030B 4E
030C 47
030D 20
030E 54
030F 4F
0310 4F
0311 20
0312 4C
0313 4F
0314 4E
0315 C7
0316 00
10
0317 53
0318 54
0319 52
031A 49
031B 4E
031C 47
031D 20
NULL
R 600
E 601
C 602
T T 603
Y 604
P 605
E 606
607
M 608
I 609
S 610
M 611
A 612
T 613
C 614
H 615
NULL 616
O 617
U 618
T 619
620
O 621
F 622
623
S 624
T 625
R 626
I 627
N 628
G 629
630
S 631
P 632
A 633
C 634
E 635
NULL 636
S 637
T 638
R 639
I 640
N 641
G 642
643
T 644
O 645
C 646
647
L 648
C 649
N 650
G 651
NULL 652
S 653
T 654
R 655
I 656
N 657
G 658
659

10

```
031E  46                                   F     660
031F  4F                                   O     661
0320  52                                   R     662
0321  4D                                   M     663
0322  55                                   U     664
0323  4C                                   L     665
0324  41                                   A     666
0325  20                                         667
0326  54                                   T     668
0327  4F                                   O     669
0328  4F                                   O     670
0329  20                                         671
032A  43                                   C     672
032B  4F                                   C     673
032C  4D                                   M     674
032D  50                                   P     675
032E  4C                                   L     676
032F  45                                   E     677
0330  08                                   ^H    678
0331  00                                   NULL  679
```

11

```
0332  43                                   C     680
0333  41                                   A     681
0334  4E                                   N     682
0335  27                                   '     683
0336  54                                   T     684
0337  20                                         685
0338  43                                   C     686
0339  4F                                   C     687
033A  4E                                   N     688
033B  54                                   T     689
033C  49                                   I     690
033D  4E                                   N     691
033E  55                                   U     692
033F  45                                   E     693
0340  00                                   NULL  694
```

12

```
0341  55                                   U     695
0342  4E                                   N     696
0343  44                                   D     697
0344  45                                   E     698
0345  46                                   F     699
0346  49                                   I     700
0347  4E                                   N     701
0348  45                                   E     702
0349  44                                   D     703
034A  20                                         704
034B  55                                   U     705
034C  53                                   S     706
034D  45                                   E     707
034E  52                                   R     708
034F  20                                         709
0350  46                                   F     710
0351  55                                   U     711
0352  4E                                   N     712
0353  43                                   C     713
0354  54                                   T     714
0355  49                                   I     715
0356  4F                                   O     716
0357  CE 00                                N     717
0359  2C   DB   X0359  FLAG  USED BY INPUT FOR TEST AT 0849   ,     718
035A  00   1. X035A  YOP  FIRST BYTE IN BUFFER . (LINE BUFFER - USED FOR INPUT & OUTPUT  NULL  719
```

```
035B  00          2.  NOP
035C  00              NOP
035D  00              NOP      LINE BUFFER AREA
035E  00              NOP
035F  00              NOP      USED FOR INPUT + OUTPUT
0360  00              NOP
0361  00              NOP
0362  00              NOP
0363  00              NOP
0364  00              NOP
0365  00              NOP
0366  00              NOP
0367  00              NOP
0368  00              NOP
0369  00              NOP
036A  00              NOP
036B  00              NOP
036C  00              NOP
036D  00              NOP
036E  00              NOP
036F  00              NOP
0370  00              NOP
0371  00              NOP
0372  00              NOP
0373  00              NOP
0374  00              NOP
0375  00              NOP
0376  00              NOP
0377  00              NOP
0378  00              NOP
0379  00              NOP
037A  00              NOP
037B  00              NOP
037C  00              NOP
037D  00              NOP
037E  00              NOP
037F  00              NOP
0380  00              NOP
0381  00              NOP
0382  00              NOP
0383  00              NOP
0384  00              NOP
0385  00              NOP
0386  00              NOP
0387  00              NOP
0388  00              NOP
0389  00              NOP
038A  00              NOP
038B  00              NOP
038C  00              NOP
038D  00              NOP
038E  00              NOP
038F  00              NOP
0390  00              NOP
0391  00              NOP
0392  00              NOP
0393  00              NOP
0394  00              NOP
0395  00              NOP
0396  00          61. NOP
```

```
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
```

```
0397 00                62. NOP                                                                  780
0398 00                    NOP          LINE BUFFER AREA                                        781
0399 00                    NOP          USED FOR INPUT & OUTPUT                                 782
039A 00                    NOP                                                                  783
039B 00                    NOP                                                                  784
039C 00                    NOP                                                                  785
039D 00                    NOP                                                                  786
039E 00                    NOP                                                                  787
039F 00                    NOP                                                                  788
03A0 00                    NOP                                                                  789
03A1 00                72. NOP                                                                  790
03A2 00          X03A2     NOP PRINT SUPPRESSION FLAG  =0 PRINT  =1 NO PRINT                    791
03A3 00          X03A3     NOP BRANCHING FLAG FOR DIM ERASE LET  (0EE1)                         792
03A4 00          X03A4     NOP SIZE OF VARIABLE (TYPE CODE) FOUND IN SYMBOL TABLE (0F29)        793
03A5 00          X03A5     NOP TEMP STORAGE FOR STRING EVALUATOR (0D2F 0D79)                    794
03A6 00          X03A6     NOP } ADDR OF FIRST BYTE OF STRING SPACE (ABOVE STACK, TOP DOWN)      795
03A7 00                    NOP }                                                                796
03A8 00          X03A8     NOP } "NEW" SETS THIS PTR TO 03AA                                    797
03A9 00                    NOP } PTR TO STRING FORMULA TABLE                                    798
03AA 00          X03AA     NOP          STRING FORMULA TABLE                                    799
03AB 00                    NOP                                                                  800
03AC 00                    NOP          3 BYTES PER ENTRY IN TABLE                              801
03AD 00                    NOP            CHAR COUNT - 1 BYTE                                   802
03AE 00                    NOP            PTR TO 1ST BYTE OF CHAR STRING - 2 BYTES              803
03AF 00                    NOP                                                                  804
03B0 00                    NOP                                                                  805
03B1 00                    NOP  30 LOC'S                                                        806
03B2 00                    NOP                                                                  807
03B3 00                    NOP          IDENTITY GROUP 'STACK'                                  808
03B4 00                    NOP                                                                  809
03B5 00                    NOP          03A8 CONTAINS 'STACK PTR' VALUE                         810
03B6 00                    NOP                                                                  811
03B7 00                    NOP                                                                  812
03B8 00                    NOP                                                                  813
03B9 00                    NOP                                                                  814
03BA 00                    NOP                                                                  815
03BB 00                    NOP                                                                  816
03BC 00                    NOP                                                                  817
03BD 00                    NOP                                                                  818
03BE 00                    NOP                                                                  819
03BF 00                    NOP                                                                  820
03C0 00                    NOP                                                                  821
03C1 00                    NOP                                                                  822
03C2 00                    NOP                                                                  823
03C3 00                    NOP                                                                  824
03C4 00                    NOP                                                                  825
03C5 00                    NOP                                                                  826
03C6 00                    NOP                                                                  827
03C7 00                    NOP                                                                  828
03C8 00          X03C8     NOP CHAR COUNT OF CHAR STRING TO PRINT                               829
03C9 00          X03C9     NOP                                                                  830
03CA 00                    NOP PTR TO 1ST BYTE OF CHAR STRING TO PRINT                          831
03CB 00          X03CB     NOP                                                                  832
03CC 00                    NOP PTR TO NEXT AVAILABLE BYTE IN STRING SPACE                       833
03CD 00          X03CD     NOP TEMP STORAGE USED BY DIM & STRING EVALUATOR - also DEF evaluator  834
03CE 00                    NOP also STRING PURGE - 12D3 & 120F - FORMATTED # CONVERT BIT-FLAG    835
03CF 00          X03CF     NOP } LINE # OF 'DATA' STMT MOST RECENTLY USED BY 'READ'             836
0300 00                    NOP }                                                                837
0301 00          X0301     NOP "NEW" SETS THIS LOC TO 00  ERASE SETS THIS LOC TO 1 FOR-64       838
0302 00          X0302     NOP INPUT SETS THIS LOC TO 00  READ SETS THIS LOC TO AF              839
```

```
0303 00          X0303  NOP  } NEW SETS THIS TO 1 LESS THAN START  } ADDR OF CURRENT INSTR BEING   840 } TEMP STORAGE FOR  ← NEXT
0304 00                 NOP  } ADDR OF STMT AREA                   } EXECUTED BY BASIC          841 }
0305 00          X0305  NOP  } ADDR OF DECIMAL PT IN CHAR STRING OF CONVERTED #                  842
0306 00                 NOP  } TEMP STORAGE USED BY DIM & STRING EVALUATOR                      843
0307 00          X0307  NOP  LINE # OF STMT WHERE ERROR OCCURS  } LINE # OF CURRENT INSTR BEING  844 } COMMAND MODE SETS THIS TO FFFF
0308 00                 NOP  (SEE Ø4A6)                         } EXECUTED BY BASIC             845 } USED TO PREVENT SOME COMMANDS IN DIRECT MODE
0309 00          X0309  NOP  } LINE # OF STMT BEING EXECUTED BY BASIC WHEN STOP, END, or        846
030A 00                 NOP  } CNTL-C ENCOUNTERED                                               847
030B 00          X030B  NOP  "NEW" SETS THIS FLAG TO ØØØØ - USED BY SOME INSTR - NOT USABLE AS DIRECT COMMAN 848 BEFORE A PROGRAM HAS BEEN RUN
030C 00                 NOP  ADDR OF STMT BEING EXECUTED BY BASIC WHEN STOP, END, or CNTL-C     849 ENCOUNTERED
030D F5          X030D  PUSH PSW } ADDR OF START OF STACK - SET BY INITIALIZATION                850
030E 29                 DAD  H  }                                                               851
030F 00          X030F  NOP      Contains Addr of First byte in stmt area (PTR TO LO-BYTE OF     852 CHAIN ADDR OF FIRST STMT
03E0 00                 NOP                                              AFTER 'NEW' THIS PTR   853 IS 2 LESS THAN THAN THE VALUE IN THE NXT 3 FF
03E1 00          X03E1  NOP      Ptr to Next available byte in stmt area (PTS TO WHERE LO-        854 BYTE OF PACKED-FORMAT LINE #
03E2 00                 NOP  ALSO - START ADDR OF SYMBOL TABLE)          IS TO BE STORED        855 FOR NEXT STMT ENTERED
03E3 00          X03E3  NOP  "NEW" SETS THIS TO NXT AVAILABLE BYTE ADDR                          856
03E4 00                 NOP  ADDR OF START OF TABLE FOLLOWING SYMBOL TABLE (=ARRAY TABLE)        857
03E5 00          X03E5  NOP  "NEW" SETS THIS TO NXT AVAILABLE BYTE ADDR                          858
03E6 00                 NOP  ADDR OF BYTE FOLLOWING END OF ARRAY TABLE                          859
03E7 00          X03E7  NOP  "NEW" SETS THIS TO 1 LESS THAN START ADDR OF STMT AREA              860
03E8 00                 NOP  PTR FOR NXT DATA STMT WHILE PROGRAM IS RUNNING                     861
03E9 00          X03E9  NOP  WHEN ERROR OCCURS, STORE REG E VALUE - IF Ø2 (SYNTAX ERROR) TRIGGERS CALL 862 TO EDIT ROUTINE
03EA 00          X03EA  NOP  THIS AREA CHANGED TO Ø4 BY "NEW"                                    863
03EB 00                 NOP                                                                     864
03EC 00                 NOP    26 LOC - CORRESPONDS TO LETTERS 'A-Z'                            865
03ED 00                 NOP    SEE Ø F21                                                        866
03EE 00                 NOP                                                                     867
03EF 00                 NOP  THIS IS THE VARIABLE DEFAULT TABLE                                 868
03F0 00                 NOP                                                                     869
03F1 00                 NOP  FIRST LOC CORRESPONDS TO VARIABLE NAMES STARTING                   870 WITH AN 'A'
03F2 00                 NOP  LAST LOC - 'Z'                                                     871
03F3 00                 NOP                                                                     872
03F4 00                 NOP  NEW SETS ALL 26 LOC TO Ø4 - THIS IS THE                            873
03F5 00                 NOP                                                                     874
03F6 00                 NOP  "SIZE" OF SINGLE PRECISION VARIABLES                               875
03F7 00                 NOP                                                                     876
03F8 00                 NOP                                                                     877
03F9 00                 NOP                                                                     878
03FA 00                 NOP                                                                     879
03FB 00                 NOP                                                                     880
03FC 00                 NOP                                                                     881
03FD 00                 NOP                                                                     882
03FE 00                 NOP                                                                     883
03FF 00                 NOP                                                                     884
0400 00                 NOP                                                                     885
0401 00                 NOP                                                                     886
0402 00                 NOP                                                                     887
0403 00                 NOP                                                                     888
0404 00          X0404  NOP  } 8 BYTE HOLDING AREA (USED BY SWAP)                               889
0405 00                 NOP  }                                                                  890
0406 00                 NOP  }                                                                  891
0407 00                 NOP  }                                                                  892
0408 00                 NOP  }                                                                  893
0409 00                 NOP  }                                                                  894
040A 00                 NOP  }                                                                  895
040B 00                 NOP  }                                                                  896
040C 00          X040C  NOP  TRACE FLAG  TRACE OFF IF Ø  TRACE ON IF AF                         897
040D 00          X040D  NOP  LOST-BIT BYTE FOR DOUBLE-PREC RIGHT SHIFT (see 1F44) FOR Ø4BE-Ø4I5 898
040E 00          X040E  NOP  # HOLDING AREA, LSB FOR DOUBLE PRECISION VALUES                     899
```

```
                                # HOLDING AREA
040F  00              NOP                                                    900
0410  00        X0410 NOP                                                    901
0411  00        X0411 NOP                                                    902
0412  00        X0412 NOP  } INTEGER } SINGLE                                903
0413  00              NOP  } VALUE   } PRECISION                             904
0414  00        X0414 NOP            } VALUE                                 905
0415  00        X0415 NOP                                                    906
                                                  DOUBLE PRECISION VALUE
                                0415 = EXPONENT BYTE FOR SINGLE OR DOUBLE PREC FLOATING POINT #'S
0416  00        X0416 NOP STORES SIGN BIT OF RESULT (see 1873)               907
0417  00        X0417 NOP LOST-BIT BYTE FOR 0418-041F                        908
0418  00        X0418 NOP TEMP STORAGE FOR A DOUBLE PREC # (see 1891)        909
0419  00              NOP                                                    910
041A  00              NOP                                                    911
041B  00              NOP                                                    912
041C  00              NOP                                                    913
041D  00              NOP                                                    914
041E  00        X041E NOP                                                    915
041F  00        X041F NOP                                                    916
0420  00        X0420 NOP UNDER SOME SPECIAL CASES, IF OVERFLOW OF FORMATTED OUTPUT FIELD, PUT '%' CHAR 917
                                HERE
0421  00        X0421 NOP BUFR AREA FOR # CONVERSION ROUTINE 2181            918
0422  00              NOP                                                    919
0423  00        X0423 NOP  0421 IS USUALLY 1ST CHAR OF BUFR                  920
0424  00              NOP  0420 COULD BE 1ST CHAR IF OVERFLOW OF FORMATTED OUTPUT 921
0425  00              NOP        (see 2200 or 22F9).                         922
0426  00              NOP                                                    923
0427  00              NOP                                                    924
0428  00              NOP                                                    925
0429  00              NOP                                                    926
042A  00              NOP                                                    927
042B  00              NOP                                                    928
042C  00              NOP                                                    929
042D  00              NOP                                                    930
042E  00              NOP                                                    931
042F  00              NOP                                                    932
0430  00              NOP                                                    933
0431  00              NOP                                                    934
0432  00              NOP                                                    935
0433  00              NOP                                                    936
0434  00              NOP                                                    937
0435  00              NOP                                                    938
0436  00              NOP                                                    939
0437  00              NOP                                                    940
0438  00              NOP                                                    941
0439  00              NOP                                                    942
043A  00              NOP                                                    943
043B  00        X043B NOP  } TEMP STORAGE FOR DOUBLE-PRECISION #             944
043C  00              NOP  } HOLDS MULTIPLIER FOR DBL-PREC MULTIPLY          945
043D  00              NOP  }                                                 946
043E  00        X043E NOP  } HOLDS DIVIDEND FOR DBL-PREC DIVIDE              947
043F  00              NOP  }                                                 948
0440  00              NOP  }                                                 949
0441  00        X0441 NOP  }                                                 950
0442  00        X0442 NOP  }                                                 951
0443  20        X0443 DATA A' '                          space               952  CHAR STRING
0444  49              MOV  C,C                           I                   953    ␣IN␣
0445  4E              MOV  C,M                           N                   954
0446  A0              ANA  B                             space               955
0447  00        X0447 NOP                                null                956
0448  0D        X0448 DCR  C                             CR                  957  CHAR STRING
0449  0A              LDAX B                             LF                  958  CR LF OK CR LF
044A  4F              MOV  C,A                           O                   959
```

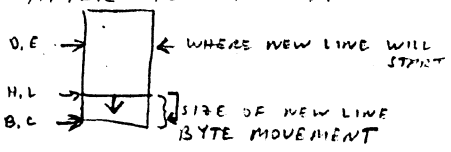
```
044B          DATA H'C9'                                  K      960
044C  00      DCR  C                                      CR     961
044D  0A      LDAX B                                      LF     962
044E  00      NOP                                         NULL   963
044F  00      X044F DCR  C                                CR     964   CHAR STRING
0450  0A      LDAX B                                      LF     965
0451  42      MOV  B,D                                    B      966   CR LF BREAK
0452  52      MOV  D,D                                    R      967
0453  45      MOV  B,L                                    E      968
0454  41      MOV  B,C                                    A      969
0455  CB      DATA H'CB'                                  K      970
0456  00      NOP                                         NULL   971
0457  21 04 00  X0457 LXI  H,X0004  IF GOSUB or FOR GROUP IS ON STACK, COMPUTE ADDR IN HL  !    972
045A  39              DAD  SP       OF THE GOSUB or FOR TOKEN                              9    973
045B  7E        →X045B MOV  A,M     FETCH BYTE FROM STACK AREA (POSSIBLY 'GOSUB' OR 'FOR' TOKEN)  974
045C  23               INX  H       ADVANCE HL TO PT TO LINE # OF GOSUB STORED ON STACK   #    975
045D  FE 81            CPI  H'81'  } RETURN IF BYTE IS NOT                                      976
045F  C0               RNZ         } A 'FOR' TOKEN (PROCESSING A 'RETURN')                @    977
0460  4E               MOV  C,M    } PUT PTR TO VARIABLE VALUE REGION IN BC               N    978
0461  23               INX  H      } (MOST RECENT ACTIVE 'FOR' VARIABLE)                  @    979
0462  46               MOV  B,M    }                                                      F    980
0463  23               INX  H      }                                                      #    981
0464  E5               PUSH H      SAVE PTR TO SIGN-OF-STEP ON STACK                          982
0465  69               MOV  L,C    } PUT PTR TO VARIABLE VALUE REGION IN HL (VARIABLE IN MOST RECENT ACTIVATED 'FOR' INSTRUCTION)  983
0466  60               MOV  H,B    }                                                          984
0467  7A               MOV  A,D    } TEST IF DE IS 0000 - DE IS 0000 ONLY IF PROCESSING A 'NEXT' INSTRUCTION  985
0468  B3               ORA  E      } DE IS PTR TO VARIABLE VALUE REGION (see 09F7) OF VARIABLE IN 'FOR' STMT BEING PROCESSED  986
0469  EB               XCHG        PUT PTR TO VAR IN CURRENT INSTR IN HL, PUT PTR TO VAR (FROM STACK) IN DE   987
046A  CA 6F 04         JZ   X046F  JMP IF PROCESSING A 'NEXT' INSTRUCTION                     J    988
046D  EB               XCHG        PUT PTR TO VAR (FROM STACK) IN HL, PUT PTR TO VAR IN CURRENT INSTR IN DE   989
046E  E7               RST  4      TEST IF GROUP OF BYTES ON STACK IS THE SAME VARIABLE AS CURRENT 'FOR' INSTR  990
046F  01 0E 00  →X046F LXI  B,X000E LOAD BC WITH 14 DECIMAL (# TO FIND EARLIER 'FOR' TOKEN ON STACK  991  (GROUP CONSISTS
0472  E1               POP  H      PUT PTR TO SIGN-OF-STEP IN HL, DE HAS PTR TO VAR VALUE (FROM STACK)  992  OF 17 BYTES)
0473  C8  ← 'NEXT' or 'FOR' with var match found on stack  RZ  RETURN IF PROCESSING 'FOR' + ENTRY ON STACK MATCHES OR PROCESSING A 'NEXT' INSTRUCTION  993
0474  09               DAD  B      LOOK AT EARLIER ENTRY ON STACK FOR VAR PTR TO MATCH VAR PTR OF CURRENT  994
0475  C3 5B 04         JMP  X045B  LOOP UNTIL NO MORE 'FOR' GROUPS ON STACK                   C    'FOR' 995
0478  CD 94 04  X0478 CALL X0494  RET HERE ONLY IF ENOUGH MEMORY FOR LINE IN BUFR TO BE          996  STORED IN STMT AREA
0478  C5        X0478 PUSH B  } SWAP BC & HL                                                   E    997
047C  E3               XTHL   } BC = new-nxt-byte                                                  998
047D  C1               POP  B } HL = former nxt byte                                           A    999
047E  E7         →X047E RST  4  TEST FOR PTR EQUALITY (RT WHEN HL = DE)                            1000
047F  7E               MOV  A,M } MOVE 1 BYTE INTO HIGHER MEMORY                                   1001
0480  02               STAX B   }                                                                  1002
0481  C8               RZ       RETURN IF HL = DE (LAST BYTE HAS BEEN MOVED)           H       1003
0482  0B               DCX  B   } ADJUST PTRS TO MOVE THE NEXT BYTE                                1004
0483  2B               DCX  H   }                                                      +       1005
0484  C3 7E 04         JMP  X047E MOVE ANOTHER BYTE                                   C       1006
0487  E5        X0487 PUSH H     SAVE PTR IN STACK                                             1007
0488  2A E5 03         LHLD X03E5  FETCH PTR STORED IN 03E5 (ADDR OF BYTE AFTER LAST BYTE  1008  OF ARRAY TABLE)
048B  06 00            MVI  B,H'00' } MULTIPLY VALUE IN C REG BY 2                              1009
048D  09               DAD  B        } AND ADD TO HL                                           1010
048E  09               DAD  B        }                                                         1011
048F  CD 94 04         CALL X0494  TEST IF ENOUGH MEMORY IS AVAILABLE                 M       1012
0492  E1               POP  H      RESTORE PTR, FETCH FROM STACK                              1013
0493  C9               RET                                                            I       1014
0494  D5        X0494 PUSH D  save ptr for nxt byte in stmt area (bufr stmt will start at this addr)  1015  SAVE DE IN STACK
0495  EB              XCHG        D,E = addr at end of stmt area after bufr inserted         1016
0496  21 D2 FF        LXI  H,XFFD2 } TEST THAT NEW-NXT-BYTE IS LESS THAN                 !R      1017
0499  39              DAD  SP      } (SP) - 46 (STACK NEEDS 46 LOC'S NOT AVAILABLE       9       1018
049A  E7              RST  4       }                                                             1019
                                    TO STMT AREA)
```

STACK

PTR TO GOSUB INSTRUCTION	← GOSUB
LINE # OF GOSUB	
GOSUB TOKEN	← SP+4
CNTL-C CHK ADDR	
RETURN ADDR FOR CALL 0457	← SP

[OPEN N BYTES IN MEMORY FOR NEW STMT TO BE INSERTED]

AFTER B,C & H,L SWAPPED



43

```
0490  EB          XCHG   } RESTORE ADDR of byte to put bufr into stmt area
049C  01          POP  D }
049D  D0          RNC    RNC IF HL ≥ DE (RNC IF SP-46₁₀ ≥ new end of stmt area)
```

E	ERROR MESSAGE
Ø1	NEXT WITHOUT. FOR
Ø7	OUT OF MEMORY
ØA	REDIMENSIONED ARRAY
Ø8	DIVISION BY ZERO
12	UNDEFINED USER FUNCTION
Ø2	SYNTAX ERROR

```
049E  1E 07       X049E  MVI  E,H'07' } ERROR MESSAGE "OUT OF MEMORY"
04A0  C3 87 04           JMP  X0487
04A3  2A CF 03    X04A3  LHLD X03CF } SAVE ADDR OF LINE # WHERE ERROR OCCURS
04A6  22 D7 03           SHLD X03D7 }
04A9  1E 02       X04A9  MVI  E,H'02'
04AB  01 1E 08           LXI  B,X081E MVI E,H'Ø8'
04AE  01 1E 01           LXI  B,X011E MVI E,H'Ø1'
04B1  01 1E 0A           LXI  B,X0A1E MVI E,H'ØA'
04B4  01 1E 12           LXI  B,X121E MVI E,H'12'
04B7  CD CC 05    X0487  CALL X05CC RESET PTRS (PART OF NEW)
04BA  AF                 XRA  A    } CLEAR PRINT SUPPRESSION FLAG
04BB  32 A2 03           STA  X03A2 }
04BE  CD 98 0A           CALL X0A98 OUTPUT CR LF & NULLS
04C1  21 19 02           LXI  H,X0219 HL POINT TO BYTE BEFORE START OF ERROR MESSAGE ASCII TABLE
04C4  7B                 MOV  A,E  } SAVE ERROR TOKEN - IF Ø2 - EXECUTE EDIT ROUTINE
04C5  32 E9 03           STA  X03E9 }                  IF "SYNTAX ERROR" SEE Ø4E8
04C8  CD 8E 09    X04C8  CALL X098E SCAN ERROR MESSAGE STRING TIL NULL FOUND
04CB  1D                 DCR  E DECREMENT ERROR MESSAGE COUNTER
04CC  23                 INX  H ADVANCE PTR TO POINT TO IST BYTE OF NXT ERROR MESSAGE
04CD  C2 C8 04           JNZ  X04C8 JMP IF NULL-COUNT NOT ZERO YET
04D0  CD 41 12    X04D0  CALL X1241 PRINT THE ASCII STRING,
04D3  2A D7 03           LHLD X03D7 FETCH LINE #
04D6  7C                 MOV  A,H  } IF HL = FFFF ZERO FLAG SET
04D7  A5                 ANA  L    } (CURRENT ADDR LOC USED AS FLAG FOR CONDITIONAL CALL - see Ø4FD)
04D8  3C                 INR  A    }
04D9  C4 6B 21           CNZ  X216B PRINT ASCII STRING " IN␣XXXXX" XXXXX = LINE #
04DC  3E C1              MVI  A,H'C1'
04DE  AF          X04DE  XRA  A      CLEAR ACC
04DF  32 A2 03           STA  X03A2   CLEAR Ø3A2 TO ØØ
04E2  21 48 04           LXI  H,X0448 LOAD HL WITH LOC OF  CR LF "OK" CRLF
04E5  CD 90 27           CALL X2790 INITIALIZATION SUBR (AFTER INITIALIZATION THIS IS CHANGED
04E8  3A E9 03           LDA  X03E9 } EXECUTE EDIT ROUTINE IF LAST ERROR MESSAGE
04EB  D6 02              SUI  H'02' } WAS "SYNTAX ERROR"
04ED  CC FE 16           CZ   X16FE
04F0  21 FF FF    X04F0  LXI  H,X~~0000~~FFFF } SET Ø3D7 & Ø3D8 TO FFFF } USED TO PREVENT INPUT
04F3  22 D7 03     X04F3 SHLD X03D7 }          } AS A DIRECT COMMAND
04F6  CD 94 06           CALL X0694 FILL INPUT BUFR WITH LINE FROM INPUT CONSOLE
04F9  D7                 RST  2 MOVE H,L TO POINT TO FIRST NON-SPACE CHAR IN INPUT BUFR
04FA  3C                 INR  A } TEST THIS FIRST NON-SPACE CHAR FOR BEING A NULL
04FB  3D                 DCR  A }
04FC  CA F0 04           JZ   X04F0 DO NOT PROCESS LINE IN BUFFER IF ALL SPACES (IST NON-SPACE
04FF  F5                 PUSH PSW SAVE CARRY FLAG  C=#  NC=CHAR (NC IF IMMEDIATE INSTR
0500  CD F3 08           CALL X08F3 REMOVE & CONVERT LINE #, HL PT TO NEXT NON-SPACE CHAR
0503  D5          EDIT→X0503 PUSH D   SAVE LINE # IN STACK
0504  CD EF 05           CALL X05EF CONDENSE LINE IN BUFFER, HL RESET TO PT TO BUFR
0507  47                 MOV  B,A  ZERO B REG, C REG IS CHAR COUNT
0508  D1                 POP  D    RESTORE LINE # TO D,E
0509  F1                 POP  PSW  RESTORE CARRY FLAG
050A  D2 33 07           JNC  X0733 EXECUTE IMMEDIATE INSTRUCTION
050D  D5                 PUSH D BUFR LINE # TO STACK
050E  C5                 PUSH B   SAVE CHAR COUNT OF CONDENSED BUFR
050F  D7                 RST  2   } FIND & TEST IST BYTE IN CONDENSED BUFR
0510  B7                 ORA  A   } (TEST FOR NULL)
0511  F5                 PUSH PSW SAVE FLAGS
0512  CD 7B 05           CALL X057B Test Stmt Area to determine where in stmt area to
0515  C5                 PUSH B   PUT ADDR OF LO BYTE OF CHAIN ADDR IN STACK (PTR TO WHERE LINE
0516  DC 39 15           CC   X1539 CARRY SET IF BUFR LINE # = STMT AREA LINE #
                                  REMOVE LINE FROM STMT AREA, ADDR OF IST BYTE FOR NEW LINE IN STACK
```

```
1020
1021  C  RESTORE DE FROM STACK
1022  P
1023
1024  C7
1025  *O
1026  *W
1027
1028
1029
1030
1031
1032  ML
1033  /
1034  2"
1035  M   := NULL
1036  !
1037
1038  2
1039  M
1040
1041  #
1042  BH
1043  MA
1044  *W
1045
1046  %
1047  <
1048  O !
1049  >A
1050  /
1051  2"
1052  !H
1053  M  'TO
1054  :
1055  V
1056  L
1057  !
1058  "W
1059  M
1060  W
1061  <
1062  =
1063  J
1064  INSTR
1065  M
1066  U
1067  M
1068  G
1069  Q
1070
1071  R3
1072  U
1073  E
1074  W
1075  7
1076
1077  M
1078  E
1079  \9
```

ERROR MESSAGE PROCESSING

LOAD E REGISTER WITH A NUMBER

BASIC COMMAND MODE PROCESSING

CALL X1241)

CHAR IS A NULL)

EXECUTE, NO LINE #)

IN BUFR

START -1

install line from condensed bufr

FROM BUFR IS TO BE STORED IN STMT AREA

(FORMER LINE MUST BE REMOVED BEFORE INSERTING NEW LINE)

44

```
                    A B C D
0519  D1                  POP  D    PTR TO ADDR TO PUT BUFR IN  ) STMT AREA
051A  F1                  POP  PSW  PUT ADDR OF CHAIN ADDR IN D,E (EITHER END OF STMT AREA OR
051B  CA 40 05            JZ   X0540 null (line # Ø was typed) reset ptrs + flags as part of
051E  2A E1 03            LHLD X03E1  H,L = addr of nxt-byte in stmt area
0521  E3                  XTHL swap HL & BC  put addr of nxt byte in stack, B is Ø   C is byte count of line in
0522  C1                  POP  B    nxt byte addr in B,C
0523  09                  DAD  B    } calculate increase of memory used in stmt area, put new addr of
0524  E5                  PUSH H    }
0525  CC 7A 04            CALL X0478 SHUFFLE STMT AREA TO OPEN SPACE FOR LINE IN BUFR
0528  F1                  POP  H    }
0529  22 E1 03            SHLD X03E1 } ADJUST PTR TO NXT-BYTE AVAILABLE IN STMT AREA
052C  EB                  XCHG HL= ADDR IN STMT AREA FOR BUFR LINE, DE = NXT BYTE AVAILABLE (AFTER
052D  74                  MOV  M,H  ???
052E  23                  INX  H    } ADJUST PTR TO PT TO LO-BYTE OF LINE # POSITION IN
052F  23                  INX  H    }       STMT AREA
0530  D1                  POP  D    LINE # OF LINE IN BUFR INTO D,E
0531  73                  MOV  M,E  }
0532  23                  INX  H    } STORE BUFR LINE # IN STMT AREA
0533  72                  MOV  M,D  } ADJUST PTR TO POINT TO ASCII AREA OF STMT
0534  23                  INX  H    }
0535  11 5A 03            LXI  D,X035A DE POINT TO FIRST BYTE IN CONDENSED BUFR
0538  1A          →X0538   LDAX D    FETCH A BYTE FROM THE CONDENSED BUFR
0539  77                  MOV  M,A  STORE THE BYTE IN STMT AREA
053A  23                  INX  H    INCR STMT AREA PTR
053B  13                  INX  D    INCR BUFR PTR
053C  B7                  ORA  A    TEST BYTE JUST MOVED
053D  C2 38 05            JNZ  X0538 MOVE ANOTHER BYTE IF NOT A NULL
0540  CD A7 05    X0540   CALL X05A7 RESET EXECUTION FLAGS + PTRS (PART OF "NEW")
0543  23                  INX  H    HL POINTS   START OF STMT AREA
0544  EB                  XCHG      PUT HL IN DE TO ENTER LOOP
0545  62          →X0545  MOV  H,D  } ADJUST H,L TO POINT TO FIRST BYTE OF LINE FOR
0546  68                  MOV  L,E  }    HL TO SCAN
0547  7E                  MOV  A,M  } TEST BOTH BYTES OF CHAIN ADDR IN STMT AREA
0548  23                  INX  H    }
0549  B6                  ORA  M    }
054A  CA F0 04            JZ   X04F0 IF NULL, ENTIRE STMT AREA PROCESSED. FETCH ANOTHER
054D  23                  INX  H    } ADVANCE HL TO POINT TO BYTE BEYOND LINE #
054E  23                  INX  H    } IN STMT AREA
054F  23                  INX  H    }
0550  AF                  XRA  A    CLEAR ACC TO ØØ
0551  BE          →X0551  CMP  M    } SCAN FOR NULL, HL PT TO FIRST BYTE OF NXT
0552  23                  INX  H    } LINE WHEN NULL IS FOUND
0553  C2 51 05            JNZ  X0551 }
0556  EB                  XCHG      SAVE PTR TO CHAIN ADDR LO BYTE IN DE
0557  73                  MOV  M,E  } STORE CHN ADDR AT FRONT 2 BYTES OF PREVIOUS
0558  23                  INX  H    }   LINE IN STMT AREA
0559  72                  MOV  M,D  }
055A  C3 45 05            JMP  X0545 REPEAT FOR NXT LINE
055D  11 00 00    X055D   LXI  D,X0000 } PUT LN # ØØØØ ON STACK
0560  D5                  PUSH D    }
0561  CA 6F 05  RSRVD WRD JZ   X056E JMP IF RESERVED WRD FOLLOWED BY A ':' or a null
                NO ARGS
0564  D1                  POP  D    REMOVE ADDR FROM STACK
0565  CC F3 08            CALL X08F3 ASSEMBLE NXT CHAR(S) INTO A PACKED 2-BYTE VALUE -
0568  D5                  PUSH D    PUT LINE # ON STACK (1ST ARG)
0569  CA 77 05  RSRVD WRD JZ   X0577 IF NON-SPACE CHAR AFTER LAST DIGIT IS A or colon NULL, JMP
                1 ARG
056C  CF                  RST  1    } TEST NEXT CHAR FOR SYNTAX ERROR, MUST BE A '-'
056D  AF                  XRA  A    } THIS BYTE IS THE HEX CODE FOR THE RESERVED WRD '-'
056E  11 FA FF    X056E   LXI  D,XFFFA DELETE WITH NO ARG, PUT LINE # 65530 ON STACK, SKIP NXT
0571  C4 F3 08            CNZ  X08F3 CONVERT SECOND NUMERIC ARGUMENT TO 2 BYTES IN DE
0574  C2 A9 04            JNZ  X04A9 SYNTAX ERROR IF SECOND NUMERIC ARGUMENT NOT FOLLOWED BY
                                A NULL OR COLON
```

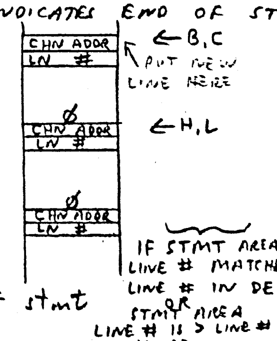
1080 WHERE IN MIDDLE OF STMT AREA CONDENSED
1081 LINE MUST BE WEDGED IN
1082 "NEW"
1083
1084 condensed buFr } HL = char count
1085 A } BC = addr of
1086 new nxt-available byte in stack nxt-byte
1087
1088
1089
1090
1091 ADJUSTMENT)
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115 LINE OR COMMAND FROM TTY
1116
1117
1118 CHN ADDR LO ← D,E
HI
LINE # LO
HI
1119
1120
1121
1122
1123
1124 CHN ADDR LO ← H,L D,E
HI
LINE # LO
HI
1125
1126
1127
1128
1129
1130
1131
1132
1133 IGNORE SPACES
1134 (only 1 numerical argument to instruction
1135
1136
1137 INSTRUCTIONS
1138
1139

AFTER AFTER
UP TO 0556 Ø556 (XCHG) Ø546 (LOOP)
H,L

D,E
H,L

LOOP MOVES H,L TO NXT NULL +1

```
                                                                     BEFORE   STK
                                                                              RET ADDR
                                                                              LINE #
0577  EB          X0577  XCHG       } SWAP RET ADDR + LINE #
0578  D1                 POP   D    } ON STK                    Q     AFTER   STK
0579  E3                 XTHL       }                                         LINE #
057A  E5                 PUSH  H    }    LINE # IN DE   HL DON'T CARE           STK
-------------------------------------------------------------------------------------------
057B  2A DF 03    X057B  LHLD  X03DF  H,L <- FIRST ADDR IN STMT AREA   D+E = LINE # of line in  *_   FIND STMT IN STMT AREA
057E  44        ->X057E  MOV   B,H  } SAVE ADDR IN B,C (TEMPORARY STORAGE)  condensed buffer    D
057F  4D                 MOV   C,L  }                                                           M
0580  7E                 MOV   A,M    LOAD LO BYTE OF CHAIN ADDR              } TEST FOR   2 NULLS IN A ROW
0581  23                 INX   H      ADVANCE STMT AREA PTR                   }          #  INDICATES END OF STMT AREA
0582  B6   CLEARS CARRY  ORA   M      OR HI BYTE OF CHAIN ADDR INTO ACC       }          6
0583  2B                 DCX   H      MOVE STMT AREA PTR BACK 1               }          +
0584  C8                 RZ           IF CHAIN ADDR IS ØØ IN STMT AREA, RETURN}          H
0585  23                 INX   H    } ADVANCE STMT AREA PTR TO POINT TO LO BYTE OF LINE #  #
0586  23                 INX   H    }                                                      #
0587  7E                 MOV   A,M  }
0588  23                 INX   H    } PUT LINE # (16 BIT VALUE) IN H+L                     #
0589  66                 MOV   H,M  } (LINE # of line in stmt area)
058A  6F                 MOV   L,A  }
                                              RNZ
058B  E7                 RST   4      RZ HL=DE   RC HL<DE    RNC HL≥DE
058C  60                 MOV   H,B  } ADDR IN STMT AREA (PTS TO LO BYTE OF CHAIN ADDR) ie 1st byte of stmt
058D  69                 MOV   L,C  } BC=HL'S
058E  7E                 MOV   A,M  }                                ADDR OF 1ST
058F  23                 INX   H    } PUT CHAIN ADDR IN H,L (BYTE OF NXT LINE IN               #
0590  66                 MOV   H,M  }                                STMT AREA)
0591  6F                 MOV   L,A  }
0592  3F                 CMC                                                                    ?
0593  C8                 RZ           RETURN IF LINE # OF LINE IN CONDENSED BUFR = LINE # POINTED  H  TO BY B,C (START H,L AT ADDR
                                             CARRY SET IF RZ ie HL=DE                                       IN Ø3DF + Ø3EØ)
0594  3F                 CMC                                                                    ?
0595  D0                 RNC          RETURN IF LINE # OF LINE IN CONDENSED BUFR < LINE # POINTED  P  TO BY B,C
0596  C3 7E 05           JMP   X057E  TEST NEXT LINE IN STMT AREA                              C
-------------------------------------------------------------------------------------------
NEW -> 0599  C0          RNZ          IF "NEW" TOKEN FOLLOWED BY ANY CHAR NOT A NULL, COLON, SPACE - WRONG SYNTAX    NEW
059A  2A DF 03    X059A  LHLD  X03DF  FETCH ADDR OF START OF STMT AREA INTO HL              *_
059D  CD 3F 08           CALL  X083F  SETS LOC Ø40C TO ØØ - TURN TRACE FLAG OFF             M?
05A0  77                 MOV   M,A  } PUT A DOUBLE NULL IN FIRST TWO LOC'S IN STMT AREA      } ELIMINATE PROGRAM BY RESETTING
05A1  23                 INX   H    } ADVANCE PTR BY 2                                     #  } FLAGS FOR STMT AREA
05A2  77                 MOV   M,A  }                                                         }
05A3  23                 INX   H    }                                                      #  }
05A4  22 E1 03           SHLD  X03E1  RESET END-OF-STMT-AREA PTR TO 2 LARGER THAN START ADDR "  }
-------------------------------------------------------------------------------------------
05A7  2A DF 03    X05A7  LHLD  X03DF } SET ADDR'S Ø3D3 + Ø3D4 TO POINT TO ADDR           *_   ENTER HERE - RUN WITH
05AA  2B                 DCX   H     } ONE LESS THAN START OF STMT AREA (START PROG AT 1ST STMT)  +   NO ARGUMENT
05AB  22 D3 03    X05AB  SHLD  X03D3 }                                                      "S
05AE  3E 1A              MVI   A,H'1A' USE ACC AS A LOOP CTR                               >
05B0  21 EA 03           LXI   H,X03EA USE HL AS A PTR                                     !
05B3  36 04     ->X05B3  MVI   M,H'04' } PUT Ø4 IN LOC Ø3EA - Ø403 (26 LOC TOTAL)          6
05B5  23                 INX   H       } RESET DEFAULT TABLE TO SINGLE PRECISION           #
05B6  3D                 DCR   A       } SIZE (4 BYTES PER VARIABLE)                       =
05B7  C2 B3 05           JNZ   X05B3   }                                                   B3
05BA  2A A6 03           LHLD  X03A6 } ELIMINATE ALL STRINGS STORED IN STRING SPACE        *&
05BD  22 CB 03           SHLD  X03CB }                                                     "K
05C0  CD E1 07           CALL  X07E1 RESET PTR FOR NXT (OR CURRENT) DATA STMT TO START OF PGM  M — SAME AS RESTORE
05C3  2A E1 03           LHLD  X03E1 } RESET THESE PTRS TO POINT TO FIRST BYTE AVAILABLE   *
05C6  22 E3 03           SHLD  X03E3 } AFTER STMT AREA                                     "
05C9  22 E5 03           SHLD  X03E5 } - CLEAR SYMBOL TABLE + ARRAY TABLE                  "
05CC  C1          X05CC  POP   B     REMOVE ADDR OR RETURN ADDR FROM STACK                A
05CD  2A DD 03           LHLD  X03DD } SET STACK PTR TO START ADDR OF STACK STORED IN Ø3DD  *}
05D0  F9                 SPHL        }
05D1  21 AA 03           LXI   H,X03AA } RESET PTR TO FIRST ADDR OF TABLE                  !*
05D4  22 A8 03           SHLD  X03A8   }                                                   "(
05D7  AF                 XRA   A     CLEAR ACC                                             /
05D8  67                 MOV   H,A   CLEAR H
```



IF ALL LINE #'S IN STMT AREA ARE LESS THAN LINE # IN DE, BC + HL POINT TO END OF STMT AREA

ON RETURNING FROM THIS ROUTINE B,C PT TO FIRST BYTE OF LINE WITH LINE # ≥ LINE # IN DE

```
05D9                    MOV  L,A  CLEAR L                                          1200
05DA 22 00 03           SHLD X0300 CLEAR PTR TO 0000                        "(     1201
05DD 32 01 03           STA  X0301 CLEAR LOC 0301 TO 00 - CLEAR FLAG LOC    2Q     1202
05E0 E5                 PUSH H  PUT 0000 ON STACK                                  1203
05E1 C5                 PUSH B  PUT RETURN ADDR BACK ON STACK               E      1204
05E2 2A 03 03  X05E2    LHLD X0303 RESTORE H,L TO PT TO ONE LESS THAN START OF STMT AREA*S  1205
05E5 C9                 RET                                                 I      1206
-------------------------------------------------------------------------------------------
05E6 3E 3F     X05E6    MVI  A,A'?'  } OUTPUT "?"                           >?     1207   OUTPUT "?␣"
05E8 DF                 RST  3       }                                      _      1208
05E9 3E 20              MVI  A,A' '  } OUTPUT "␣"                           >      1209
05EB CF                 RST  3       }                                      _      1210   INPUT FROM TTY
05EC C3 94 06           JMP  X0694 JMP TO TTY INPUT ROUTINE                 C      1211
-------------------------------------------------------------------------------------------
05EF AF        X05EF    XRA  A       } CLEAR TEMP STORAGE BYTE              /      1212
05F0 32 A5 03           STA  X03A5   }                                      2%     1213
05F3 0E 05              MVI  C,H'05' START CHAR COUNT AT 5 (2 FOR CHN ADDR, 2 FOR PACKED  1214  LINE #, 1 FOR NULL AT END)
05F5 11 5A 03           LXI  D,X035A START PTR AT FIRST CHAR IN INPUT BUFR  Z      1215
05F8 7E        X05F8    MOV  A,M  FETCH CHAR FROM BUFR                             1216
05F9 FE 20              CPI  A' '    } IF CHAR IS A SPACE, PUT IN CONDENSED BUFR    1217
05FB CA 4D 06           JZ   X064D   }                                      JM     1218
05FE 47                 MOV  B,A  SAVE CHAR IN B REG, IF CHAR IS NULL CLEAR B REG  G  1219
05FF FE 22              CPI  A'"'    } IF CHAR IS " , PUT CHAR IN CONDENSED BUFR TIL NEXT  1220
0601 CA 60 06           JZ   X0660   }                                      J      1221
0604 B7                 ORA  A       } IF CHAR IS NULL, END OF PROCESSING TO CONDENSE  7  1222
0605 CA 91 06           JZ   X0691   }                                      J      1223   BUFR
0608 3A A5 03           LDA  X03A5   } RELOAD ACC FROM TEMP STORAGE  {03A5 ≠ 0 IF PROCESSING  2%  1224
0608 B7                 ORA  A       }                                A DATA STMT   J    1225
060C 47                 MOV  B,A  SAVE CHAR IN B                    G      1226
060D 7E                 MOV  A,M  FETCH SAME CHAR AGAIN FROM BUFR                  1227
060E C2 4D 06           JNZ  X064D JMP IF (03A5) ≠ 0 KEEP PUTTING CHAR IN CONDENSED BUFR  BM  1228  TIL NULL OR COLON FOUND
0611 FE 3F              CPI  A'?'    } IF CHAR IS '?' [PRINT] PUT CODE BYTE IN      ?      1229
0613 3E A0              MVI  A,H'A0' }                                      >      1230
0615 CA 4D 06           JZ   X064D   } CONDENSED BUFR                       JM     1231
0618 7E                 MOV  A,M  FETCH CHAR FROM BUFR AGAIN                       1232
0619 FE 30              CPI  A'0'    } TEST + JMP IF CHAR IS LESS THAN A '0'        0      1233
061B DA 23 06           JC   X0623   }                                      ZD     1234
061E FE 3C     X061E    CPI  A'<'    } TEST CHAR, IF DIGIT 0-9, or ':' ';' PUT BYTE IN  <  1235
0620 DA 4D 06           JC   X064D   } CONDENSED BUFR                       ZM     1236
-------------------------------------------------------------------------------------------
0623 D5        X0623    PUSH D  SAVE PTR TO WRD IN BUFR, TRY TO DETERMINE IF WRD IS A  U  1237  RESERVED WRD
0624 11 79 00           LXI  D,X0079 USE DE TO PT TO RESERVED WRD LIST, HL PTS TO WRD IN BUFR  1238
0627 E5                 PUSH H  SAVE PTR TO CONDENSED BUFR, SKIP RST 2 TO GET INTO LOOP  1239
0628 3E 07     DUMMY BYTE  MVI  A,H'07'                                    >H     1240
062A 13                 INX  D  ADVANCE RESERVED WRD PTR TO NEXT CHAR IN WRD       1241
062B 1A        X062B    LDAX D  FETCH RESERVED WRD CHAR                            1242
062C E6 7F              ANI  H'7F'   } TEST CHAR FROM RESERVED WRD LIST, IF END OF LIST JMP  1243
062E CA 3F 06           JZ   X063F   }                                      J?     1244
0631 BE                 CMP  M  COMPARE RESERVED WRD CHAR WITH CHAR OF WRD IN BUFR  >  1245
0632 C2 74 06           JNZ  X0674 NO MATCH, MOVE PTRS TO LOOK AT NXT RESERVED WRD  B  1246
0635 1A                 LDAX D       } FETCH RESERVED WRD CHAR                     1247
0636 B7                 ORA  A       } TEST D7 BIT                          7      1248
0637 F2 29 06           JP   X0629   } IF D7 BIT IS 0, NOT THE LAST CHAR OF RESERVED, TRY TO  )  1249  MATCH NXT CHAR
063A F1                 POP  PSW  RESERVED WRD TOTAL MATCH, POP PSW REMOVES PTR (ON STACK) TO START OF  1250  WRD IN BUFR JUST MATCHED
063B 78                 MOV  A,B  PUT RESERVED WRD FAIL-TO-MATCH CTR IN ACC        1251
063C F6 80              ORI  H'80' ADD 80 TO FORM CODE BYTE FOR RESERVED WRD (ORI SETS FLAG  1252  TO SKIP NEXT 2 INST + DUMMY BYTE)
063E F2 E1 7E  POP H MOV A,M JP  X7EE1 RESTORE HL TO PT TO BUFR, FETCH CHAR FROM BUFR  1253
0641 D1                 POP  D  RESTORE DE TO POINT TO CONDENSED BUFR       Q      1254
-------------------------------------------------------------------------------------------
0642 E8                 XCHG  EXCHANGE REGISTERS                                   1255
0643 FE 90              CPI  H'90'   } IF RESERVED WRD IS 'ELSE', PUT ':' IN CONDENSED BUFR  1256
0645 36 3A              MVI  M,A':'  }                                      6:     1257   THEN PUT ELSE TOKEN = 90H
0647 C2 4C 06           JNZ  X064C                                          BL     1258
064A 0C                 INR  C  ADVANCE CHAR COUNT                                 1259
```

```
0649 23          INX  H ADVANCE PTR TO CONDENSED BUFR                          #    1260
064C EB       X064C XCHG   EXCHANGE REGISTERS AGAIN; HL PT TO BUFR, DE PT TO CONDENSED  1261 BUFR
064D 23       >X064D INX  H ADVANCE PTR TO LOOK AT NXT CHAR IN BUFR            #    1262
064E 12          STAX D STORE CHAR IN BUFR                                          1263
064F 13          INX  D ADVANCE CONDENSED BUFR PTR                                  1264
0650 0C          INR  C INCREMENT CHAR COUNT                                        1265
0651 FE 3A       SUI  A':'  } IF CHAR IS A COLON - STORE IN CONDENSED BUFR-FETCH NXT V: CHAR 1266
0653 CA 5B 06    JZ   X065B }                                                  JC   1267
0656 FE 49       CPI  A'I'  } IF TOKEN (KEYWORD CODE) IS 83 = 'DATA', STORE NON-ZERO # IN I03A5 1268 (NOTE: CPI DOES NOT ZERO ACC.)
0658 C2 5E 06    JNZ  X055E                                                    0^   1269
065B 32 A5 03  >X065B STA  X03A5 SET 03A5 TO NON-ZERO IF DATA STMT, SET BACK TO ZERO AT 2% NXT 1270 ':' FOUND
065E D6 54       X065E SUI  A'T' } IF TOKEN IS 8E = 'REM'                      VT   1271
0660 C2 F8 05          JNZ  X05F8 IF NOT REM - WORK ON NXT CHAR IN LINE BUFR   B    1272
0663 47                MOV  B,A PUT CHAR IN B (REM WILL GET STUCK IN NXT LOOP TIL NULL) G 1273
0664 7E          X0664 MOV  A,M } IF CHAR IS A NULL, END OF PROCESSING TO CONDENSE BUFR  1274
0665 B7                ORA  A  }                                               7    1275
0666 CA 81 06          JZ   X0681 }                                            J    1276
0669 B8                CMP  B  TEST CHAR FOR 2ND "                             8    1277
066A CA 4D 06          JZ   X064D                                              JM   1278
066D 23          X0660 INX  H  ADVANCE PTR TO BUFR                            #    1279
066E 12                STAX D  STORE CHAR IN CONDENSED BUFR                        1280
066F 0C                INR  C  INCR CHAR COUNT                                     1281
0670 13                INX  D  ADVANCE CONDENSED BUFR PTR                          1282
0671 C3 64 06          JMP  X0664 STAY IN THIS LOOP TIL 2ND " FOUND, OR IF REM, MOVE REST OF CBUFR 1283 INTO CONDENSED BUFR TIL NULL FOUND
0674 E1          X0674 POP  H } RESET PTR TO POINT TO FIRST CHAR OF WRD IN BUFR FOR NXT  1284 ATTEMPTED RESERVED WRD MATCH
0675 E5                PUSH H }                                                    1285
0676 04                INR  B ADVANCE RESERVED WRD FAIL-TO-MATCH CTR               1286
0677 EB                XCHG EXCHANGE REGISTERS SO HL POINT TO RESERVED WRD LIST     1287
0678 B6          >X0678 ORA  M FETCH CHAR OF RESERVED WRD                    6    1288
0679 23                INX  H ADVANCE PTR TIL POINTING TO FIRST BYTE OF NXT RESERVED # WRD 1289 LIST
067A F2 78 06          JP   X0678 LAST CHAR OF RESERVED HAS 07 SET (JMP MINUS)     1290
067D EB                XCHG EXCHANGE REGISTERS AGAIN                               1291
067E C3 20 06          JMP  X0620 TEST NXT RESERVED FOR A MATCH                C+  1292
0681 21 59 03  >X0681  LXI  H,X0359 RESET PTR TO POINT AT BYTE BEFORE START OF BUFR !Y 1293
0684 12                STAX D }                                                    1294
0685 13                INX  D } STORE 3 NULLS AT END OF CONDENSED BUFR             1295
0686 12                STAX D }                                                    1296
0687 13                INX  D }                                                    1297
0688 12                STAX D }                                                    1298
0689 C9                RET    END OF STMT PROCESSING                           I   1299
-------------------------------------------------------------------------------------------------------------------------------------
068A 05 DEL CHAR >X068A DCR  B } DELETE 1 CHAR (SUBTRACT 1 FROM CHAR COUNT & LINE BUFR PTR) 1300 GET CHAR FROM
068B 2B                 DCX  H }                                                +   1301
068C DF                 RST  3  ECHO CHAR (BACKARROW)                           _   1302 INPUT CONSOLE
068D C2 99 06           JNZ  X0699                                              B   1303
0690 DF END OF LINE >X0690 RST  3  ECHO CHAR                                    _   1304 & PROCESS IT
0691 CD 99 0A           CALL X0A99 PRINT CRLF + NULLS                           M   1305
-------------------------------------------------------------------------------------------------------------------------------------
0694 21 5A 03    X0694  LXI  H,X035A SET HL TO POINT TO FIRST BYTE OF LINE BUFR   !Z  1306
0697 06 01              MVI  B,4'01' INITIALIZE CHAR CTR TO 1                       1307
0699 CD EC 06  >X0699   CALL X06EC  GET A CHAR FROM KEYBOARD                    M   1308
069C FE 07              CPI  H'07' } TEST FOR CHAR = CNTL G (BELL)                  1309
069E CA DA 06           JZ   X06DA }                                            J:  1310
06A1 FE 0D              CPI  H'0D' } CHAR = CR                                      1311
06A3 CA 93 0A           JZ   X0A93 }                                            J   1312
06A6 FE 20              CPI  A' '  } IF ANY OTHER CNTL CHAR, IGNORE IT              1313
06A8 DA 99 06           JC   X0699 }                                            Z   1314
06AB FE 7D              CPI  H'7D' } IF ALT MODE CHAR, IGNORE IT                    1315
06AD D2 99 06           JNC  X0699 }                                            R   1316
06B0 FE 40              CPI  A'@'  } TERMINATES LINE INPUT & CANCELS LINE IN BUFR @ 1317
06B2 CA 90 06           JZ   X0690 }                                            J   1318
06B5 FE 5F              CPI  A'_'  } IF CHAR IS A BACKARROW, DELETE CHAR        _   1319
```

```
06B7  CA 8A 06          JZ    X068A   } IF CHAR IS A BACKARROW, DELETE IT
06BA  4F          X060A MOV   C,A     SAVE CHAR IN C
06BB  78                MOV   A,B     }
06BC  FE 48             CPI   A'H'    } IF B>72 (CHAR COUNT IS > SIZE OF LINE BUFR)
06BE  3E 07             MVI   A,H'07' } RING BELL & NO MORE CHAR INTO 'LINE BUFR
06C0  D2 C7 06          JNC   X06C7   }
06C3  79                MOV   A,C     PUT CHAR BACK INTO ACC
06C4  71                MOV   M,C     STORE CHAR IN LINE BUFR
06C5  23                INX   H       INCR LINE BUFR PTR
06C6  04                INR   B       INCR CHAR COUNT
06C7  DF          X06C7 RST   3       ECHO CHAR ON OUTPUT CONSOLE
06C8  C3 99 06          JMP   X0699   FETCH ANOTHER CHAR
```

```
06CB  C2 72 12    X06CB JNZ   X1272   IF PRINT SUPPRESSION FLAG SET (LOC Ø3A2), JMP, POP CHAR OFF STACK & RET
06CE  F1                POP   PSW     } PUT CHAR IN ACC & LEAVE CHAR ON STACK
06CF  F5                PUSH  PSW     }
06D0  FE 20             CPI   A' '    TEST FOR CONTROL CHAR - IF CHAR IS, PRINT IMMEDIATELY
06D2  DA E1 06          JC    X06E1   CONTROL CHAR ARE NUMERICALLY LESS THAN A SPACE
06D5  3A 27 00          LDA   X0027   = CHAR COUNT FOR LINE ALREADY PRINTED
06D8  FE (48)           CPI   A'H'    } IF COUNT IS ALREADY = TERMINAL WIDTH, PRINT CR LF & NULLS
06DA  CC 98 0A          CZ    X0A98   }
06DD  3C                INR   A       INCREMENT CHAR COUNT
06DE  32 27 00          STA   X0027   STORE IT IN Ø027
06E1  DB 00       X06E1 IN    H'00'   } TEST OUT PUT CONSOLE STATUS
06E3  E6 80             ANI   H'80'   }
06E5  C2 E1 06          JNZ   X06E1   LOOP UNTIL READY
06E8  F1                POP   PSW     } FETCH CHAR FROM STACK & PRINT IT
06E9  D3 01             OUT   H'01'   } IF Ø3A2 = 1 NO PRINTING
06EB  C9                RET
```

TERMINAL WIDTH

PRINT AN ASCII CHAR

CHAR TO PRINT IS ON STACK

RST 3 falls into this routine

```
06EC  DB 00       X06EC IN    H'00'   } TEST INPUT CONSOLE STATUS
06EE  E6 01             ANI   H'01'   }
06F0  C2 EC 06          JNZ   X06EC   LOOP UNTIL DATA AVAILABLE
06F3  DB 01             IN    H'01'   } INPUT DATA (1 CHAR) & STRIP OFF THE D7 PARITY BIT
06F5  E6 7F             ANI   H'7F'   }
06F7  FE 0F             CPI   H'0F'   } RETURN HERE UNLESS CONTROL O CHAR
06F9  C0                RNZ           }
06FA  3A A2 03          LDA   X03A2   } IF CONTROL O CHAR, COMPLEMENT SUPPRESS PRINT FLAG
06FD  2F                CMA           }
06FE  32 A2 03          STA   X03A2   }
0701  C9                RET
```

INPUT A CHAR FROM INPUT CONSOLE

RETURN WITH CHAR IN ACC

```
0702  3E E4             MVI   A,H'64' } STORE 64 AT LOC Ø301
0704  32 01 03          STA   X0301   }
0707  CD AB 09          CALL  X09AB   FIRST PART OF INSTRUCTION SAME AS LET
070A  E3                XTHL  REMOVE CNTL-C CHK ADDR FROM STACK, PUT PTR TO INSTRUCTION ON STACK
070B  CD 57 04          CALL  X0457   TEST STACK CONTENTS TO SEE IF A 'FOR' GROUP OF BYTES USING THE SAME VAR EXISTS
070E  D1                POP   D       PUT PTR TO INSTRUCTION IN DE
070F  C2 14 07          JNZ   X0714   JMP IF ACTIVE 'FOR' GROUPS ON STACK DO NOT USE THE SAME VARIABLE
0712  09                DAD   B       } ADD 14 DECIMAL TO HL, VAR IN ACTIVE 'FOR' GROUP ON STACK MATCHED, CANCEL ANY
0713  F9                SPHL          } GROUPS LOWER ON STACK, SP RESET TO REINITIALIZE 'FOR' GROUP, SAME POSITION ON STACK
0714  EB          X0714 XCHG  PUT PTR TO INSTRUCTION IN HL
0715  0E (08)           MVI   C,H'08' } TEST IF ENOUGH SPACE BETWEEN END OF ARRAY TABLE + CURRENT END
0717  CD 97 04          CALL  X0497   } OF STACK FOR 16 MORE BYTES = 2x # IN CREG (ONLY IF BYTES NEEDED)
071A  E5                PUSH  H       PUT PTR TO INSTRUCTION ON STACK
071B  CD AC 09          CALL  X09AC   SCAN 'FOR' INSTRUCTION TO FIND ADDR OF TERMINATOR (NULL OR ':')
071E  E3                XTHL  PUT PTR TO INST IN HL, PUT 'FOR' TERMINATOR ADDR ON STACK
071F  E5                PUSH  H       } PUT LINE # OF 'FOR' INSTRUCTION ON STACK
0720  2A D7 03          LHLD  X03D7   } LEAVE PTR TO INSTRUCTION IN HL (SHOULD POINT TO 'TO' TOKEN)
0723  E3                XTHL          }
0724  CF                RST   1       } SYNTAX CHECK FOR 'TO' TOKEN
0725  A7          DB 'A7' ANA   A       }
0726  F7                RST   6       TEST VALUE-TYPE OF VARIABLE USED AS CTR IN 'FOR' INSTRUCTION
```

ERROR CHANGE TO Ø9

DB 'A7'

FOR

FOR {VAR.} = {EXPRESSION} TO {EXPRESSION} STEP {EXPRESSION}

OPTIONAL DEFAULTS TO 1

STACK

ADDR OF NULL OR ':' AT END OF 'FOR' INSTRUCTION
LINE # OF 'FOR'
2ND EXPRESSION
3RD EXPRESSION (STEP SIZE)
(TYPE - 3) OF VAR IN 'FOR' INSTR
SIGN OF STEP
PTR TO VAR VALUE REGION
'FOR' TOKEN

TOTAL : 17 BYTES

```
0727  CA 8A 1C          JZ    X1C8A IF VARIABLE IS STRING TYPE
072A  D2 8A 1C          JNC   X1C8A IF VARIABLE IS DBL-PREC TYPE } PRINT ERROR MESSAGE 'TYPE MISMATCH'
072D  F5                PUSH  PSW SAVE FLAG STATUS ON STACK (VARIABLE IS INTEGER-RM, SNG-PREC - RPO)
072E  CD 52 0C          CALL  X0C52 EVALUATE 2ND EXPRESSION (AFTER 'TO' TOKEN)
0731  F1                POP   PSW FETCH FLAG STATUS FROM STACK
0732  E5                PUSH  H SAVE PTR TO INSTRUCTION ON STACK.
0733  F2 4C 07          JP    X074C JMP IF VARIABLE IS SNG-PREC TYPE, 'FOR' INSTRUCTION WILL USE SNG-PREC ARITHMETIC
0736  CD 11 1C          CALL  X1C11 CONVERT VALUE IN HOLDING AREA (2ND EXPRESSION) TO INTEGER FORMAT
0739  E3                XTHL  PUT PTR TO INSTRUCTION IN HL, PUT INTEGER VALUE OF 2ND EXPRESSION ON STACK
073A  11 01 00          LXI   D,X0001 LOAD DE WITH DEFAULT STEP SIZE OF 1 (INTEGER FORMAT)
073D  7E                MOV   A,M FETCH NXT CHAR FROM INSTRUCTION STRING
073E  FE AD             CPI   H'AD' } IF NXT CHAR IS A 'STEP' TOKEN, STARTING AT THE NXT CHAR AFTER TOKEN,
0740  CC 67 14          CZ    X1467 } EVALUATE 3RD EXPRESSION (STEP SIZE) TO AN INTEGER VALUE
0743  D5                PUSH  D PUT STEP SIZE ON STACK
0744  E5                PUSH  H PUT PTR TO INSTRUCTION ON STACK
0745  EB                XCHG  PUT STEP SIZE IN HL FOR NXT TEST, PTR TO INSTRUCTION IN DE
0746  CD 2F 19          CALL  X192F TEST STEP SIZE FOR ZERO + SIGN
0749  C3 60 07          JMP   X0760
074C  CD 45 1C  X074C    CALL  X1C45 CONVERT VALUE IN HOLDING AREA (2ND EXPRESSION) TO SNGPREC FORMAT USING CSGN ROUTINE
074F  CD 51 1B          CALL  X1B51 PUT FLOATING-PT VALUE FOR 2ND EXPRESSION IN BCDE, B=EXPONENT BYTE
0752  E1                POP   H PUT PTR TO INSTRUCTION IN HL
0753  C5                PUSH  B } PUT FLOATING-PT VALUE OF 2ND EXPRESSION ON STACK
0754  D5                PUSH  D }
0755  01 00 81          LXI   B,X8100 } LOAD BCDE WITH DEFAULT STEP SIZE OF 1 (SNG-PREC FORMAT 81 00 00 00)
0758  51                MOV   D,C }
0759  5A                MOV   E,D }
075A  7E                MOV   A,M FETCH NXT CHAR FROM INSTRUCTION STRING
075B  FE AD             CPI   H'AD' TEST IF NXT CHAR IS A 'STEP' TOKEN
075D  3E 01             MVI   A,H'01' SET SIGN OF STEP TO 1 (POSITIVE)
075F  C2 6E 07          JNZ   X076E JMP IF NO 'STEP' TOKEN FOUND
0762  CD 53 0C          CALL  X0C53 EVALUATE 3RD EXPRESSION (STEP SIZE) TO # IN HOLDING AREA
0765  E5                PUSH  H PUT PTR TO INSTRUCTION ON STACK
0766  CD 45 1C          CALL  X1C45 CONVERT VALUE IN HOLDING AREA (2ND EXPRESSION) TO SNG-PREC FORMAT USING CSGN ROUTINE
0769  CD 51 1B          CALL  X1B51 LOAD BCDE WITH VALUE FOR 2ND EXPRESSION
076C  EF                RST   5 TEST STEP SIZE FOR ZERO & SIGN
076D  E1        X0760    POP   H PUT PTR TO INSTRUCTION IN HL
076E  C5        X076E    PUSH  B } IF SNG-PREC VARIABLE BEING PROCESSED, THIS PUTS THE STEP SIZE ON THE STACK
076F  D5                 PUSH  D } IF INTEGER VARIABLE, IRRELEVANT, PUT 4 BYTES ON STACK FOR ALIGNMENT PURPOSES
0770  4F                MOV   C,A PUT SIGN OF STEP IN REG C (-1, Ø, 1 => NEG, Ø, POS)
0771  F7                RST   6 TEST VALUE-TYPE OF STEP, PUT (TYPE-3) IN ACC
0772  47                MOV   B,A PUT (TYPE-3) OF STEP IN B
0773  C5                PUSH  B PUT (TYPE-3), THEN SIGN OF STEP ON STACK
0774  E5                PUSH  H } PUT PTR TO VARIABLE VALUE REGION ON STACK
0775  2A 03 03          LHLD  X0303 } (LET PUTS ADDR IN Ø303)
0778  F3                XTHL
0779  06 81   X0779     MVI   B,H'81' } PUT 1 BYTE 'FOR' TOKEN ON STACK
077B  C5                PUSH  B
077C  33                INX   SP } NOTE: B IS PUT ON STACK 1ST, INX SP CANCELS 2ND BYTE PUT ON STACK
077D  DB 00     X077D    IN    H'00' INPUT TTY STATUS WORD
077F  E6 01             ANI   H'01' TEST FOR CHAR HAVING BEEN TYPED
0781  CC F0 07          CZ    X07F0 IF A CHAR WAS TYPED, FETCH CHAR FROM INPUT CONSOLE, RETURN IF CHAR NOT A CNTL-C
0784  22 03 03          SHLD  X0303 SAVE PTR TO CURRENT INSTR (USED BY CONT)
0787  7E                MOV   A,M FETCH NXT CHAR FROM STMT
0788  FE 3A             CPI   A':' } JMP IF CHAR IS A ':' - EXECUTE NXT INSTRUCTION IN STMT
078A  CA 83 07          JZ    X0783 }
078D  B7                ORA   A } OTHERWISE, IF CHAR IS NOT A NULL, PRINT ERROR MESSAGE "SYNTAX ERROR"
078E  C2 A9 04          JNZ   X04A9 }
0791  23                INX   H } TEST NXT 2 BYTES AFTER NULL TO SEE IF DOUBLE NULL
0792  7E                MOV   A,M } (TEST FOR END OF STMT AREA)
0793  23                INX   H }
```

```
1380
1381
1382
1383
1384
1385
1386
1387 USING CINT ROUTINE
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398 USING CSGN ROUTINE
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430 IF CHAR NOT A CNTL-C
1431
1432
1433
1434
1435
1436
1437
1438
1439
```

STACK

	INTEGER	SNG-PREC
	'FOR' TERMINATOR ADDR	
	'FOR' LINE #	
Q	2ND EXPRESSION	2ND EXPRESSION
Z	3RD EXPRESSION	
	4 bytes	3RD EXPRESSION
-	(TYPE-3) = FF	(TYPE-3) = Ø1
>	SIGN OF STEP	SIGN OF STEP
B	PTR TO VAR VALUE REGION	
MS	'FOR' TOKEN	

TOTAL = 17 BYTES FOR EACH ACTIVE 'FOR'

CNTL-C CHECK

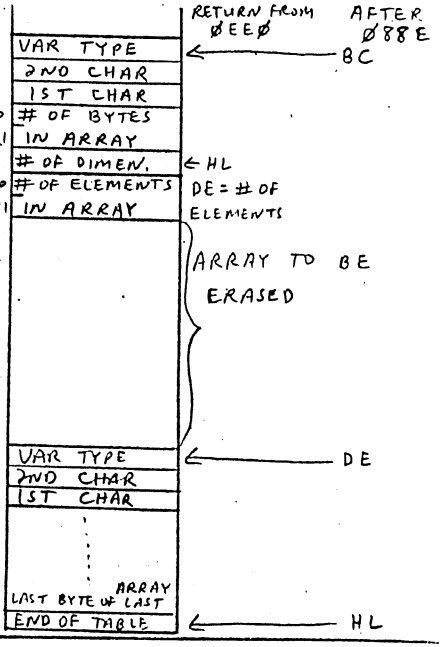
```
                        } TEST 2 BYTES AFTER NULL
0794  B           ORA   M } FOR DOUBLE NULL                                  6    1440
0795  23          INX   H ADVANCE STMT PTR TO POINT TO FIRST BYTE OF LINE # OF NXT STMT  1441
0796  CA FC 07    JZ    X07FC JMP IF END OF STMT AREA (DO A 'STOP' - TERMINATE PROGRAM)  J    1442
0799  5E          MOV   E,M } FETCH LINE # OF NXT STMT INTO DE               ^    1443
079A  23          INX   H   }                                                #    1444
079B  56          MOV   D,M }                                                V    1445
079C  EB          XCHG      } STORE LINE # AT LOC'S Ø3D7 & Ø3D8, PTR TO STMT IN DE  1446
079D  22 D7 03    SHLD  X03D7}                                               "W   1447
07A0  3A 0C 04    LDA   X040C FETCH TRACE FLAG                               :    1448
07A3  B7          ORA   A   } IF TRACE FLAG IS Ø, EXECUTE NEXT INSTRUCTION   7    1449
07A4  CA B2 07    JZ    X07B2} TRACE IS OFF IF FLAG = Ø                      J2   1450
07A7  D5          PUSH  D SAVE PTR TO STMT ON STACK                          U    1451
07A8  3E 5B       MVI   A,A'[' } PRINT '[' CHAR                              >(   1452
07AA  DF          RST   3     }                                              _    1453
07AB  CD 73 21    CALL  X2173 CONVERT LINE # (IN HL) TO DECIMAL ASCII & PRINT M !  1454
07AE  3E 5D       MVI   A,A']' } PRINT ']' CHAR                              >}   1455
07B0  DF          RST   3     }                                              _    1456
07B1  D1          POP   D FETCH PTR TO STMT FROM STACK                       Q    1457
07B2  EB       X07B2 XCHG  PUT PTR TO STMT BACK IN HL                              1458
-------------------------------------------------------------------------------------------
07B3  D7       X07B3 RST   2 SCAN CONDENSED BUFR, RZ IF NULL or colon        W    1459 COMPUTE SUBROUTINE ADDR
07B4  11 7D 07       LXI   D,X077D } CNTL C CHECK                                1460 OF INSTRUCTION & EXECUTE
07B7  D5             PUSH  D      }                                         U    1461
07B8  C8       X07B8 RZ    Return if colon pointed to by H,L                 H    1462 INSTRUCTION BY BRANCHING
07B9  D6 80    X07B9 SUI   H'80' SUBTRACT CODE WRD OFFSET                    V    1463 TO THAT SUBROUTINE
07BB  DA AB 09       JC    X09AB JMP IF NOT A RESERVED WRD, JMP TO LET ROUTINE Z+   1464
07BE  FE 26          CPI   A'&' CODE WRD A6 [TAB(] + LARGER NOT ALLOWED TO BE FIRST  &    1465 ON A LINE
07C0  D2 A9 04       JNC   X04A9 IF A6 or larger, JMP & PRINT SYNTAX ERROR     R)   1466
07C3  07             RLC     MULTIPLY BY 2 (RESERVED WRD ADDR TABLE IS 2 BYTES FOR     1467 EACH RESERVED WRD)
07C4  4F             MOV   C,A } BC = [(CODE WRD) - 80H] * 2                   O    1468
07C5  06 00          MVI   B,H'00' }                                                 1469
07C7  EB             XCHG  SAVE PTR TO INSTRUCTION STRING BEING EXECUTED, IGNORE DE  1470
07C8  21 A5 01       LXI   H,X01A5 HL = ADDR OF RESERVED WRD ADDR TABLE        !%   1471
07CB  09             DAD   B CALCULATE ADDR TO FETCH ADDR FROM TABLE                 1472
07CC  4E             MOV   C,M }                                               N    1473
07CD  23             INX   H   } FETCH ADDR FROM TABLE INTO B,C                #    1474      IF ENTER FROM HERE
07CE  46             MOV   B,M }                                               F    1475 SET CARRY or NOT CARRY
07CF  C5             PUSH  B PUT ADDR OF SUBROUTINE ON STACK                   E    1476 DEPENDENT ON FIRST NON-SPACE
07D0  EB             XCHG  HL POINTS TO INSTRUCTION STRING                           1477 BYTE AFTER RESERVED WRD IN STI
07D1  23       X07D1 INX   H   ADVANCE STRING PTR                              #    1478
07D2  7E             MOV   A,M FETCH NEXT BYTE FROM INST STRING                      1479
07D3  FE 3A          CPI   A':'                                                :    1480
07D5  D0             RNC   RNC IF ':' or larger                                P    1481
07D6  FE 20    X07D6 CPI   A' ' }                                                    1482 BYTE
07D8  CA D1 07       JZ    X07D1 } SKIP OVER ANY SPACES AFTER RESERVED WRD CODE JQ   1483
07DB  FE 30          CPI   A'0'                                                0    1484
07DD  3F             CMC                                                       ?    1485
07DE  3C             INR   A JMP TO SUBROUTINE FOUND IN ADDR TABLE             <    1486
07DF  3D             DCR   A { RNC IF NON-SPACE CHAR AFTER CODE BYTE IS NOT A  =    1487 DIGIT
07E0  C9             RET     { RC  IF NON-SPACE CHAR AFTER CODE BYTE IS A DIGIT!    1488 RZ IF CHAR IS A NULL
-------------------------------------------------------------------------------------------
07E1  EB       X07E1 XCHG      SAVE HL IN DE                                           1489 . RESTORE
07E2  2A 0F 03       LHLD  X030F LOAD START ADDR OF STMT AREA.                 *_   1490
07E5  2B             DCX   H     DECREMENT BY 1                                +    1491
07E6  22 E7 03 X07E6 SHLD  X03E7 STORE ADDR AT Ø3E7 & Ø3E8                     "    1492
07E9  EB             XCHG        RESTORE HL TO PREVIOUS VALUE                        1493
07EA  C9             RET                                                       I    1494
-------------------------------------------------------------------------------------------
07EB  DB 00    X07EB IN    H'00' INPUT TTY STATUS WRD                          {    1495
07ED  E6 01          ANI   H'01' TEST FOR CHAR HAVING BEEN TYPED                     1496
07EF  C0             RNZ   RETURN IF CHAR NOT TYPED                            @    1497
07F0  CD EC 06 X07F0 CALL  X06EC INPUT A CHAR FROM INPUT CONSOLE, CHAR IN ACC  M    1498
07F3  FE 03          CPI   H'03' IS CHAR CNTL-C ?                                    1499
```

```
07F5  C0                RNZ  IF CHAR IS NOT A CNTL-C, RETURN
07F6  F6 C0             DUMMY ORI H'C0'  RETURN TO CNTL-C CHECK IF END NOT FOLLOWED BY ':' or NULL
07F8  22 03 03          SHLD X0303  SAVE PTR TO CURRENT INSTRUCTION (USED BY CONT)
07FB  C1                POP  B  REMOVE 1 ADDR FROM STACK (Ø77D IF STOP or END, Ø784 IF CNTL-C TYPED)
07FC  F5                PUSH PSW STOP - ZERO FLAG RESET  END - ZERO FLAG SET  CNTL-C - ZERO FLAG
07FD  2A 07 03          LHLD X0307  FETCH LINE # OF CURRENT STMT
0800  7D                MOV  A,L  } IF HL = FFFF ZERO FLAG SET - 'RUN' NOT EXECUTED YET
0801  A4                ANA  H    }
0802  3C                INR  A
0803  CA 0F 08          JZ   X080F JMP IF CNTL-C TYPED BEFORE PROGRAM IS 'RUN' (WILL NOT CLEAR THE FLAGS)
0806  22 09 03          SHLD X0309 SAVE LINE # OF CURRENT STMT  (USED BY CONT)
0809  2A 03 03          LHLD X0303 } SAVE ADDR OF CURRENT INSTRUCTION IN HOLDING AREA
080C  22 0B 03          SHLD X030B } (USED BY CONT)
080F  AF                XRA  A    } CLEAR PRINT SUPPRESSION FLAG
0810  32 42 03          STA  X0342}
0813  F1                POP  PSW  RESTORE STATUS OF STATUS FLAG
0814  21 4F 04          LXI  H,X044F HL POINT TO FIRST BYTE OF "CRLF BREAK"
0817  C2 00 04          JNZ  X0400 CNTL-C or STOP, PRINT MESSAGE "BREAK IN XXXXX"  XXXXX IN BP
081A  C3 0E 04          JMP  X040E NO MESSAGE IF 'END' INSTR, RETURN TO BASIC COMMAND MODE

081D  1E 11             MVI  E,H'11' ERROR MESSAGE # IN E
081F  2A 0B 03          LHLD X030B } TEST FLAG Ø3DB & Ø3DC FOR ØØØØ
0822  7C                MOV  A,H  }
0823  B5                ORA  L
0824  CA 37 04          JZ   X0437 IF Ø3DB IS ØØØØ, PROGRAM NOT 'RUN' YET, PRINT ERROR MESSAGE
0827  EB                XCHG SAVE PTR TO NXT INSTRUCTION IN DE
0828  2A 09 03          LHLD X0309 } FETCH LINE # OF CURRENT STMT FROM HOLDING AREA
082B  22 07 03          SHLD X0307 }
082E  EB                XCHG RESTORE PTR TO NXT INSTRUCTION IN HL
082F  C9                RET

0830  CD 81 14          CALL X1481 CONVERT ARGUMENT TO 1 BYTE INTEGER IN ACC
0833  C0                RNZ  IF TERMINATOR AFTER ARGUMENT IS NOT A NULL OR ':' IGNORE INSTRUCTION
0834  3C                INR  A ADD 1 TO NULL COUNT
0835  FE 48             CPI  A'H' TEST IF NULL COUNT IS GREATER THAN TERMINAL WIDTH
0837  D2 EE 08          JNC  X08EE PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"
083A  32 26 00          STA  X0026 STORE VALUE IN NULLS LOC
083D  C9                RET

083E  3E AF  XRA A      MVI  A,H'AF' } TRACE ON   { TRACE OFF
0840  32 0C 04          STA  X040C    } SET FLAG TO AF  { SET FLAG TO ØØ
0843  C9                RET

0844  CD E0 0E          CALL X0EE0 FIND 1ST VARIABLE OR ARRAY ELEMENT IN APPROPRIATE TABLE
0847  D5                PUSH D SAVE PTR TO VARIABLE VALUE REGION ON STACK (1ST ARG)
0849  E5                PUSH H SAVE PTR TO INSTRUCTION ON STACK
084A  21 04 04          LXI  H,X0404 LOAD HL WITH ADDR OF 1ST BYTE OF AN 8 BYTE HOLDING AREA
084D  CD 66 1B          CALL X1B66 MOVE N BYTES FROM 1ST VARIABLE VALUE REGION TO HOLDING AREA
0850  2A E3 03          LHLD X03E3 } PUT PTR TO INSTRUCTION IN HL
0853  E3                XTHL       } PUT ADDR OF 1ST BYTE OF ARRAY TABLE ON STACK
0854  F7                RST  6  TEST VALUE TYPE OF 1ST ARGUMENT
0855  F5                PUSH PSW SAVE ACC ON STACK (ACC = TYPE - 3)
0856  CF                RST  1 } SYNTAX CHECK FOR COMMA BETWEEN 1ST & 2ND ARGUMENTS
0857  2C                INR  L
0858  CD E0 0E          CALL X0EE0 FIND 2ND VARIABLE OR ARRAY ELEMENT IN APPROPRIATE TABLE
085B  C1                POP  B  FETCH 1ST ARG (TYPE-3) FROM STACK INTO B REG
085C  F7                RST  6  TEST VALUE TYPE OF 2ND ARGUMENT
085D  B8                CMP  B  } TEST IF BOTH ARGS ARE THE SAME TYPE
085E  C2 9A 1C          JNZ  X1C9A } IF NOT THE SAME TYPE, JMP, PRINT ERROR MESSAGE "TYPE MISMATCH"
0861  E3                XTHL PUT ADDR OF START OF ARRAY TABLE IN HL, PUT PTR TO INSTRUCTION ON STACK
0862  EB                XCHG } SAVE PTR TO 2ND ARG VARIABLE VALUE REGION ON STACK
0863  E5                PUSH H}
0864  2A E3 03          LHLD X03E3 LOAD HL WITH ADDR OF 1ST BYTE OF ARRAY TABLE AFTER 2ND VARIABLE
0867  BF                RST  4  TEST (HL) AGAINST (DE)

@  1500  STOP
@  1501  END
"S  1502
A(D)  1503
*W  1504 RESET
    1505
    1506
$  1507
<  1508
FLAGS  1509 Ø309 & Ø3DB)
"Y  1510
*S  1511
"C  1512
/  1513
2"  1514
    1515
!O  1516
BP LINE # 1517
C^  1518
    1519  CONT
*C  1520
    1521
5  1522
E  1523 "CAN'T CONTINUE"
    1524
*Y  1525
"W  1526
    1527
I  1528  NULL
H  1529
<  1531
H  1532
R  1533
2&  1534
I  1535
>/  1536 Ø83E TR ON   Ø83F TR OFF
2  1537
I  1538
M  1539  SWAP
U  1540
    1541
    1542
A  1543
*  1544
    1545
    1546
    1547
O  1548
,  1549
M  1550
A  1551
    1552
8  1553
H  1554
    1555
    1556
    1557
*  1558 FOUND
    1559
```

```
0867  C2 EE 08          JNZ   X08EE IF (HL) ≠ (DE) THEN ARRAY TABLE WAS MOVED TO MAKE ROOM FOR A NEW ENTRY IN THE SYMBOL TABLE, PRINT
086A  D1                POP   D PUT PTR TO 2ND ARG VALUE REGION IN DE                    "ILLEGAL FUNCTION CALL"
086B  E1                POP   H } PUT PTR TO 1ST ARG VALUE REGION IN HL
086C  E3                XTHL  } LEAVE PTR TO INSTRUCTION ON STACK
086D  D5                PUSH  D SAVE PTR TO 2ND ARG VALUE REGION ON STACK
086E  CD 66 1B          CALL  X1B66 MOVE N BYTES FROM 2ND ARG TO 1ST ARG VALUE REGIONS
0871  E1                POP   H PUT PTR TO 2ND ARG VALUE REGION IN HL
0872  11 04 04          LXI   D,X0404 LOAD HL WITH ADDR OF 1ST BYTE OF TEMP. HOLDING AREA
0875  CD 66 1B          CALL  X1B66 MOVE N BYTES FROM HOLDING AREA TO 2ND ARG. VALUE REGION
0878  E1                POP   H RESTORE PTR TO INSTRUCTION IN HL
0879  C9                RET   END OF 'SWAP' PROCESSING

087A  3E 01       X087A   MVI   A,H'01' } SET TEMP STORAGE LOC TO Ø1 TO INDICATE ERASE
087C  32 01 03            STA   X0301   } IS CALLING SYMBOL TABLE SEARCH ROUTINE
087F  C3 E0 0E            JMP   X0EE0 JMP TO ROUTINE TO FIND ARRAY IN ARRAY TABLE, HL PTS TO # OF DIMENSIONS ON RETURN

0882  32 01 03    X0882   STA   X0301 CLEAR BRANCH FLAG (ACC=ØØ see 1Ø13)
0885  44                 MOV   B,H } PUT PTR TO #-OF-DIMENSIONS BYTE IN BC
0886  4D                 MOV   C,L }
0887  0B                 DCX   B  } BACK-UP BC PTR TO POINT TO VAR-TYPE BYTE OF ARRAY IN TABLE
0888  0B                 DCX   B  }
0889  0B                 DCX   B  } BC PTS TO 1ST BYTE OF ARRAY IN TABLE
088A  0B                 DCX   B  }
088B  0B                 DCX   B  }
088C  19                 DAD   D  COMPUTE IN HL ADDR OF 1ST BYTE OF NXT ARRAY IN TABLE
088D  EB                 XCHG     PUT PTR TO NEXT ARRAY IN DE
088E  2A E5 03           LHLD  X03E5 LOAD HL WITH ADDR OF BYTE AFTER END OF ARRAY TABLE *
0891  E7          X0891   RST   4  TEST IF (DE) = (HL)
0892  1A                 LDAX  D  FETCH BYTE FROM NEXT ARRAY
0893  02                 STAX  B  STORE BYTE ON TOP OF ARRAY BEING ERASED
0894  13                 INX   D  ADVANCE SOURCE PTR
0895  03                 INX   B  ADVANCE DESTINATION PTR
0896  C2 91 08           JNZ   X0891 LOOP UNTIL ALL REMAINING BYTES IN ARRAY TABLE HAVE BEEN MOVED
0899  0B                 DCX   B  ADJUST PTR TO NEW END-OF-ARRAY TABLE
089A  60                 MOV   H,B } STORE NEW ADDR FOR END OF ARRAY TABLE IN FLAG AREA
089B  69                 MOV   L,C }
089C  22 E5 03           SHLD  X03E5
089F  E1                 POP   H  FETCH PTR TO INSTRUCTION STRING FROM STACK
08A0  7E                 MOV   A,M FETCH NEXT CHAR FROM INSTRUCTION
08A1  FE 2C              CPI   A',' } IF CHAR IS NOT A COMMA, RETURN TO CNTL-C CHECK
08A3  C0                 RNZ      } IN CNTL-C CHECK CHAR MUST BE A NULL OR ':' OR SYNTAX ERROR OCCURS
08A4  D7                 RST   2  SCAN TO NEXT NON-SPACE CHAR
08A5  C3 7A 08           JMP   X087A LOOP TO ERASE ANOTHER ARRAY

08A8  1E 03              MVI   E,H'03' DEFSTR } PUT VARIABLE SIZE IN REG E
08AA  01 1E 02   Dummy   LXI   B,X021E DEFINT }
08AD  01 1E 04   Dummy   LXI   B,X041E DEFSNG }
08B0  01 1E 08   Dummy   LXI   B,X081E DEFDBL }
08B3  CD E1 08           CALL  X08E1 TEST CHAR  A-Z RNC
08B6  01 A9 04           LXI   B,X04A9 } PUT ADDR ON STACK FOR IMPLIED JMP (SYNTAX ERROR ADDR)
08B9  C5                 PUSH  B       }
08BA  D8                 RC    IF NEXT CHAR NOT A-Z, PRINT MESSAGE "SYNTAX ERROR"
08BB  D6 41              SUI   A'A' SUBTRACT OFFSET
08BD  4F                 MOV   C,A } SAVE INDEX VALUE FOR DEFAULT TABLE IN B+C
08BE  47                 MOV   B,A } C IS 1ST ARG OFFSET  B IS 2ND ARG OFFSET
08BF  D7                 RST   2  SCAN STMT FOR NEXT NON-SPACE CHAR
08C0  FE AF              CPI   H'AF' TEST CHAR FOR '-' (AF = TOKEN FOR '-')
08C2  C2 CE 08           JNZ   X08CE CLEAR ACC & PRINT "SYNTAX ERROR" IF CHAR NOT '-' (FOR 2 ARGS or PROCESS SINGLE ARG)
08C5  D7                 RST   2  SCAN FOR NEXT CHAR
08C6  CD E1 08           CALL  X08E1 TEST CHAR  A-Z - RNC   (WRONG: ONLY CHANGE FIRST ARG IN TABLE)
08C9  D8                 RC    SYNTAX ERROR IF NOT A-Z
08CA  D6 41              SUI   A'A' SUBTRACT OFFSET
08CC  47                 MOV   B,A SAVE INDEX VALUE IN B
```

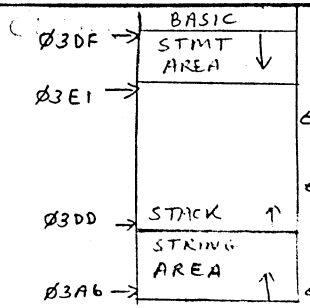
ERASE

ARRAY FOUND IN ARRAY TABLE




```
1560  "ILLEGAL FUNCTION CALL"
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1591
1582
1593
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601 DEFSTR
1602
1603 DEFINT
1604 DEFSNG
1605 DEFDBL
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
```

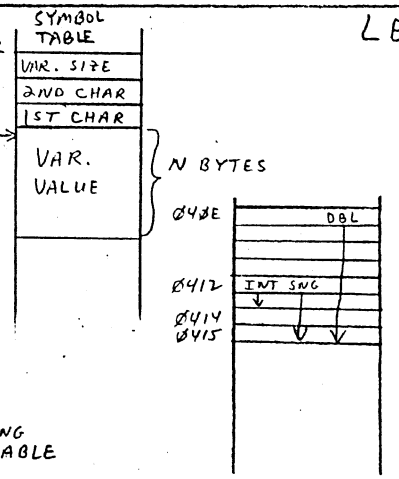
```
08CD  D7                  RST  2  SCAN FOR NEXT CHAR                                  W    1620
08CE  78           X08CE  MOV  A,B  PUT 2ND ARG OFFSET IN ACC                              1621
08CF  91                  SUB  C  SUB 1ST ARG OFFSET                                     1622
08D0  D8                  RC   SYNTAX ERROR IF 2ND IS SMALLER THAN 1ST  ie  B-A        X   1623
08D1  3C                  INR  A  ADJUST ACC TO # OF VARIABLE-LETTERS TO RE-DEFINE     <   1624
08D2  E3                  XTHL REMOVE SYNTAX ERROR ADDRESS FROM STACK, PUT PTR TO STMT ON STACK 1625
08D3  21 EA 03            LXI  H,X03EA  PUT START ADDR OF DEFAULT TABLE IN ACC        !   1626
08D6  06 00               MVI  B,H'00'  CLEAR B REG  C REG IS 1ST ARG OFFSET              1627
08D8  09                  DAD  B  COMPUTE ADDR OF 1ST-ARG VARIABLE LETTER TO BE REDEFINED 1628
08D9  73           →X08D9 MOV  M,E  STORE NEW DEFINITION SIZE IN TABLE                    1629
08DA  23                  INX  H  ADVANCE PTR TO NEXT ENTRY IN TABLE                #    1630
08DB  3D                  DCR  A  DECREMENT CTR                                     =    1631
08DC  C2 09 08            JNZ  X08D9 IF NOT ZERO - MORE ENTRIES TO BE RE DEFINED YET BY  1632
08DF  E1                  POP  H  RESTORE PTR TO STMT FROM STACK                         1633
08E0  C9                  RET  JMP TO CNTL-C CHK or ELSEWHERE                       I    1634

08E1  7E           X08E1  MOV  A,M  FETCH CHAR INTO ACC                                  1635
08E2  FE 41               CPI  A'A'  } IF CHAR IS LESS THAN                  TEST CHAR POINTED
08E4  D8                  RC         } AN 'A' RETURN WITH CARRY SET          TO BY HL      X   1637
08E5  FE 5B               CPI  A'['  } IF CHAR IS LARGER THAN A 'Z'          IF CHAR NOT A-Z  [RC 1638
08E7  3F                  CMC        } RETURN WITH CARRY SET                 SET CARRY     ?   1639
08E8  C9                  RET                                                RESET CARRY IF A-Z I RNC 1640

08E9  D7           X08E9  RST  2  SCAN STMT TO CHAR AFTER '('                        W    1641
08EA  CD 68 14     X08EA  CALL X1468 EVALUATE STRING-OBTAIN 2 BYTE INTEGER VALUE OF ARG IN DE M  1642
08ED  F0                  RP   RETURN WITH ARG IN DE ONLY IF VALUE IS ≥ Ø                 1643
08EE  1E 05        X08EE  MVI  E,H'05'  } PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"      1644 PRESENTLY SET FOR USR
08F0  C3 97 04            JMP  X0497                                                 C7   1645

08F3  2B           X08F3  DCX  H  DECREMENT PTR TO STMT-BEING-PROCESSED, THIS IS DONE TO ENTER LOOP 1646 ASSEMBLE NXT GROUP OF DIGITS
08F4  11 00 00     X08F4  LXI  D,X0000  CLEAR DE REGISTER (TEMP STORAGE)                  1647 INTO A 2 BYTE VALUE
08F7  D7           →X08F7 RST  2  POINT TO NXT NON-SPACE CHAR                        W    1648
08F8  D0                  RNC  IF CHAR IS NOT A DIGIT, RETURN PACKED LINE # IN DE      P    1649
08F9  E5                  PUSH H  SAVE STMT PTR IN STACK                                 1650
08FA  F5                  PUSH PSW  SAVE CHAR + STATUS FLAGS                             1651
08FB  21 98 19            LXI  H,X1998  LOAD H,L WITH 65529₁₀                     !    1652
08FE  E7                  RST  4  COMPARE DE + HL   RZ HL=DE  RNC HL≥DE  RC HL<DE        1653
08FF  DA A9 04            JC   X04A9  LINE # > 65529 , PRINT ERROR MESSAGE "SYNTAX ERROR" ZJ  1654
0902  62                  MOV  H,D  } SAVE # IN HL                                       1655
0903  6B                  MOV  L,E  }                                                    1656
0904  19                  DAD  D    }                                                    1657
0905  29                  DAD  H    } MULTIPLY BY 10₁₀                             )    1658
0906  19                  DAD  D    }                                                    1659
0907  29                  DAD  H    }                                              )    1660
0908  F1                  POP  PSW  RESTORE CHAR INTO ACC                                1661
0909  D6 30               SUI  A'0'  CONVERT DIGIT CHAR FROM ASCII TO HEX #        V0   1662
090B  5F                  MOV  E,A  } PUT THIS VALUE IN DE                         -    1663
090C  16 00               MVI  D,H'00'  }                                                1664
090E  19                  DAD  D  ADD TO # IN HL                                         1665
090F  EB                  XCHG  PUT # BACK IN DE                                         1666
0910  E1                  POP  H  RESTORE STMT PTR                                       1667
0911  C3 F7 08            JMP  X08F7  LOOK AT NXT CHAR IN STMT                      C    1668

0914  CA AB 05           JZ   X05AB IF NO ARGUMENT TO CLEAR, JUST DELETE ALL VARIABLES (PART OF J+"NEW" 1669
0917  CD EA 08           CALL X08EA  CONVERT ARGUMENT FROM ASCII STRING TO NUMERICAL VALUE M  1670
091A  2B                  DCX  H  BACK-UP PTR TO INSTRUCTION STRING                  +    1671
091B  D7                  RST  2  TEST TERMINATOR CHAR AT THE END OF CLEAR INSTRUCTION W  1672
091C  C0                  RNZ  DO NOT PROCESS ARGUMENT TO SET STRING SPACE IF TERMINATOR NOT Ø A NULL or ':' 1673
091D  E5                  PUSH H  SAVE PTR TO INSTRUCTION ON STACK                       1674
091E  2A A6 03            LHLD X03A6  LOAD HL WITH ADDR OF FIRST BYTE OF STRING AREA *&  1675
0921  7D                  MOV  A,L  } COMPUTE NEW ADDR FOR END OF STRING AREA            1676
0922  93                  SUB  E    } (DE) = (HL) - (DE)  -  DE HAS 2-BYTE VALUE FOR # OF 1677
0923  5F                  MOV  E,A  }                          BYTES TO RESERVE FOR CHAR 1678
0924  7C                  MOV  A,H  }                          STRINGS                   1679
```



```
0925 9                  SBC   D      } DE HAS NEW ADDR FO                    1680
0926 57                 MOV   D,A    } BYTE BELOW STRING AREA                1681
0927 DA A9 04           JC    X04A9  IF # OF BYTES SPECIFIED BY ARGUMENT IS LARGER THAN  1682 ADDR IN Ø3A6, PRINT "SYNTAX ERROR
092A 2A E1 03           LHLD  X03E1  LOAD HL WITH ADDR OF BYTE AFTER END OF STMT AREA    1683
092D 01 28 00           LXI   B,X0028} ADD 28H = 4Ø DECIMAL TO HL (ALLOW STACK SOME     1684
0930 09                 DAD   B      }                ROOM TO EXPAND                    1685
0931 E7                 RST   4  TEST (DE) AGAINST (HL)                         1686
0932 D2 9E 04           JNC   X049E IF (HL)≥(DE), STACK WOULD OVERLAP STMT AREA, SO. PRINTR  1687 MESSAGE "OUT OF MEMORY"
0935 EB                 XCHG         } STORE NEW ADDR (WHERE STACK WILL NOW START)  1688
0936 22 CD 03           SHLD  X03CD  } IN LOC Ø3DD                    "}        1689
0939 E1                 POP   H  FETCH PTR TO INSTRUCTION FROM STACK             1690
093A C3 AB 05           JMP   X05AB DELETE ALL VARIABLES AFTER SETTING AMOUNT OF STRING SPACE C+ — 1691 MOVE STACK BY CHANGING ADDR IN S
                                                                                                                    RUN
0930 CA A7 05           JZ    X05A7 IF 'RUN' WITH NO ARGUMENT, JMP TO 'NEW' SUBR —  J'START 1692 PROGRAM AT FIRST STMT
0940 CC AB 05           CALL  X05AB RESET FLAGS, THEN EXECUTE GOTO ADDR AFTER RUN TO  M+START 1693 PROGRAM
0943 01 7D 07           LXI   B,X077D} PUT A CNTL-C CHECK ADDR ON STACK FOR PSEUDO-GOTO  1694
0946 C3 59 09           JMP   X0959  } INSTRUCTION                    CY        1695
                                                                                                                    GOSUB
0949 0E 03              MVI   C,H'03'} TEST IF 6 BYTES AVAILABLE BETWEEN END OF STACK +  1696 END OF ARRAY TABLE
094B CC 87 04           CALL  X0487  }      (GOSUB ONLY NEEDS 5 BYTES)          M     1697
094E C1                 POP   B  REMOVE CNTL-C CHECK ADDR FROM STACK             A     1698
094F E5                 PUSH  H  SAVE PTR TO STMT ON STACK                       1699 AFTER Ø958
0950 E5                 PUSH  H      } PUT LINE # OF CURRENT STMT ON STACK.      1700   STACK
0951 2A 07 03           LHLD  X0307  } LEAVE PTR TO STMT IN HL                   *W    1701   PTR TO GOSUB INSTR
0954 E3                 XTHL                                                     1702   LINE # OF GOSUB
0955 16 8C              MVI   D,H'8C' PUT GOSUB TOKEN IN REG D                   1703   GOSUB TOKEN  ← STACK PTR
0957 D5                 PUSH  D  } PUT REG D ON STACK (ONLY 1 BYTE PUT ON STACK)  U     1704   CNTL-C CHECK ADDR
0958 33                 INX   SP }                                               3     1705
0959 C5          X0959  PUSH  B  PUT CNTL-C CHECK ADDR BACK ON STACK [PERFORM GOTO E NXT] 1706
                                                                                                                    GOTO
095A CD F3 08    X095A  CALL  X0RF3 ASSEMBLE NXT GROUP OF CHAR'S INTO 2 BYTE VALUE (LINE # IN H,DE)  1707
095D CD 8E 09           CALL  X098E AFTER LINE # EXTRACTED, SCAN REST OF STMT TIL NULL  MFOUND 1708
0960 E5                 PUSH  H     SAVE PTR TO NXT STMT ON STACK - PTS TO NULL AT END OF  1709 STMT
0961 2A 07 03           LHLD  X0307 FETCH LINE # OF CURRENT INSTRUCTION          *W    1710
0964 E7                 RST   4  TEST IF CURRENT INSTR LINE # = LINE # AFTER GOTO  RC HL<DE RNC HL≥DE  1711
0965 F1                 POP   H  RESTORE PTR TO NXT STMT FROM STACK              1712
0966 23                 INX   H  ADVANCE INSTRUCTION PTR TO POINT TO FIRST BYTE OF NXT INSTR AFTER GOTO  1713
0967 DC 7E 05           CC    X057E IF LINE # OF GOTO IS LARGER THAN CURRENT LINE #, SCAN STMT  1714 AREA STARTING AT CURRENT INSTR
096A D4 7B 05           CNC   X057B IF LINE # OF GOTO IS LESS THAN CURRENT LINE #, SCAN STMT  1715 AREA STARTING AT START OF STMT AREA
096D 60                 MOV   H,B} PUT ADDR OF CHN ADDR OF STMT TO GOTO IN HL    1716
096E 69                 MOV   L,C}                                               1717
096F 2B                 DCX   H  BACK-UP PTR BY 1 (ALWAYS POINT TO BYTE BEFORE INSTR WHEN +GOING  1718 TO CNTL-C CHECK)
0970 D8                 RC    RETURN TO CNTL-C CHECK ONLY IF STMT WITH MATCHING LINE # X WAS  1719 FOUND IN Ø57E or Ø578
0971 1E 08       X0971  MVI   E,H'08'} PRINT ERRDR MESSAGE "UNDEFINED STATEMENT"  1720
0973 C3 87 04           JMP   X0487  }                                C7        1721
                                                                                                                    RETURN
0976 C0                 RNZ   IGNORE RETURN IF NOT FOLLOWED BY A NULL or COLON   3     1722
0977 16 FF              MVI   D,H'FF'                                            1723 IF A GOSUB IS ON STACK
0979 CC 57 04           CALL  X0457 REMOVE A BYTE FROM STACK (AT ADDR SP+4)      MW    1724                STACK
097C F9                 SPHL  PUT ADDR OF GOSUB LINE # IN SP                     1725                PTR TO GOSUB INSTR
097D FE 8C              CPI   H'8C' IS BYTE A GOSUB TOKEN ?                      1726 after SPHL SP →LINE # OF GOSUB
097F 1E 03              MVI   E,H'03'} PRINT ERROR MESSAGE "RETURN WITHOUT GOSUB"  1727          SP+4→GOSUB TOKEN
0981 C2 87 04           JNZ   X0487  } IF BYTE IS NOT A GOSUB TOKEN              B7    1728                CNTL-C CHECK ADDR
0984 E1                 POP   H      } FETCH LINE # OF GOSUB INSTR FROM STACK    1729          SP →Ø97C
0985 22 07 03           SHLD  X0307  } STORE IT AT FLAG LOC FOR LINE # OF CURRENT INSTR "W  1730 within Ø457
0988 21 7D 07           LXI   H,X077D} FETCH PTR TO GOSUB INSTR FROM STACK INTO HL, PUT CNTL-C  1731
098B E3                 XTHL         } CHECK ADDR ON STACK, SCAN OVER REST OF GOSUB TIL NULL or ':' FOUND  1732
DATA 098C 01 3A 0E X098C LXI  B,X0E3A DATA — LXI B,XØE3A  REM — MVI C,ØØ  ELSE  :  1733 ELSE & REM + DATA
098F 00                 NOP                                                      1734
0990 0E 00              MVI   B,H'00'                                            1735 DATA  B Ø  C 3A  SCAN TIL colon or null FOUND
0992 79          X0992  MOV   A,C} SWAP B+C REG'S  AFTER SWAP                    1736
0993 48                 MOV   C,B} ACC HAS NULL                                  1737 REM  Ø Ø  NULL FOUND
0994 47                 MOV   B,A}                                    H          1738 ELSE Ø Ø  NULL FOUND
0995 7E          X0995  MOV   A,M FETCH NEXT CHAR FROM CURRENT INSTRUCTION  G     1739 Ø98E Ø Ø  NULL FOUND
```

DATA : B NULL, C NULL
REM NULL NULL
ELSE NULL NULL
Ø98E NULL NULL

```
0996  B7          ORA  A   } TEST CHAR & RETURN IF NULL (ANY OF 4 INSTR TYPES)   7   1740
0997  C9          RZ       }                                                     H   1741  WHILE EXECUTING BASIC,
0998  B8          CMP  B   } TEST CHAR IF DATA STMT, RETURN WHEN COLON FOUND     8   1742
0999  C8          RZ       }                                                     H   1743  THE CURRENT INSTRUCTION
099A  23          INX  H   ADVANCE PTR TO NXT CHAR IN INSTRUCTION                #   1744
099B  FE 22       CPI  A'"'} TEST CHAR, IF CHAR IS '"' (DATA INSTR) SCAN LINE TIL "   1745  POINTER MUST BE PUSHED
099D  CA 92 09    JZ   X0992} ANOTHER '"' FOUND' or NULL                        J   1746
09A0  D6 8A       SUI  H'8A'} JMP IF NXT CHAR IS NOT 'IF' TOKEN (8A)            V   1747
09A2  C2 95 09    JNZ  X0995} ACC=0                                             B   1748  TO LEAVE PTR AT ADDR
09A5  B8          CMP  B   FORCE CARRY BIT SET IF 'IF' TOKEN FOUND (NOT WITHIN QUOTES)  8   1749  BEFORE FIRST BYTE OF NXT
09A6  8A          ADC  D   ADD D + CARRY TO ACC                                     1750
09A7  57          MOV  D,A PUT SUM BACK IN D, (ADD 1 TO D IF 'IF' TOKEN FOUND)  W   1751  EXECUTABLE INSTRUCTION
09A8  C3 95 09    JMP  X0995 CONTINUE LOOPING TIL NULL OR COLON FOUND           C   1752
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
09AB  CD E0 0E  X09AB CALL X0EE0 FIND VARIABLE IN SYMBOL TABLE, ADD VARIABLE IF NOT FOUND, DE PT TO VALUE REGION  M  1753  LET
09AE  CF          RST  1   } TEST THAT NEXT NON-SPACE CHAR IS '='   HL PT TO STRING TO EVAL  O  1754
09AF  B9          DB '85' CMP B } IF NOT '=' CHAR, PRINT MESSAGE "SYNTAX ERROR"  8   1755
09B0  E9          XCHG     } PUT PTR TO VARIABLE VALUE REGION IN LOC'S 03D3 + 03D4   1756
09B1  22 D3 03    SHLD X03D3} AFTER: HL PTS TO '=' IN INSTRUCTION  } USED BY 'FOR'S  1757
09B4  E9          XCHG       DE PTS TO VARIABLE VALUE REGION  } see 0775            1758
09B5  C5        X09B5 PUSH D  STORE PTR TO VARIABLE REGION (IN SYMBOL TABLE) ON STACK U  1759
09B6  F7          RST  6   TEST VARIABLE TYPE (TEST LOC 03A4 + SET FLAGS)          1760
09B7  F5          PUSH PSW SAVE FLAG STATUS + ADJUSTED TYPE CODE ON STACK          1761
09B8  CD 52 0C    CALL X0C52 EVALUATE NEXT PART OF STRING - RESULT IN HOLDING AREA, TYPE FLAG IN REG A  1762
09BB  F1          POP  PSW FETCH ADJUSTED VARIABLE TYPE CODE FROM STACK            1763
09BC  E3  STORE VALUE IN  XTHL PUT PTR TO INSTRUCTION ON STACK, PUT PTR TO VARIABLE VALUE REGION IN HL  1764
09BD  C6 03 VARIABLE X09BD ADI H'03' READJUST VARIABLE TYPE CODE (COMPENSATE FOR RST 6)  F  1765
09BF  CD 90 11    CALL X1190 CONVERT VALUE IN HOLDING AREA TO FORMAT TO MATCH VARIABLE  1766
09C2  CD 98 1B  INT, SNG, or  CALL X1B98 LOAD DE WITH ADDR OF LSB OF VALUE IN HOLDING AREA (DE=0412 or HL=04BE)  1767
09C5  E5          DBL  PUSH H  SAVE PTR TO VARIABLE VALUE AREA (IN SYMBOL TABLE) ON STACK  1768
09C6  C2 F4 09    JNZ  X09F4 JMP IF INT, SNG, or DBL VARIABLE TYPE (FLAGS SET BY RST 6 IN B1B98)  1769
09C9  2A 12 04    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF IDENTITY GROUP IN STRING  1770  STRING VARIABLE
09CC  E5          PUSH H   PUT ADDR OF CHAR COUNT BYTE ON STACK  FORMULA TABLE     1771
09CD  23          INX  H   ADVANCE PTR TO STRING FORMULA TABLE - OR STRING VAR IN SYMBOL OR ARRAY TABLES  1772
09CE  5E          MOV  E,M } PUT PTR TO 1ST CHAR OF STRING VALUE (RESULT OF 0C52 AT 09BB)  1773
09CF  23          INX  H   } IN DE                                                 #   1774
09D0  56          MOV  D,M }                                                       V   1775  RNE HL ≥ DE
09D1  21 A2 03    LXI  H,X03A2 LOAD HL WITH ADDR OF BYTE AFTER END OF LINE BUFFER  !"  1776
09D4  E7          RST  4   TEST PTRS AGAINST EACH OTHER, TELLS WHERE STRING IS LOCATED  1777
09D5  D2 E8 09    JNC  X09E8 JMP IF STRING IS  IN LINE BUFR AREA (HL ≥ DE) - REMOVE IDENTITY  17.78  GROUP FROM STRING STACK
09D8  2A C0 03    LHLD X03C0 LOAD HL WITH ADDR OF BYTE 1 LESS THAN START OF STRING AREA*  1779
09DB  E7          RST  4   TEST PTRS AGAINST EACH OTHER                            1780
09DC  D1          POP  D   PUT ADDR OF CHAR COUNT BYTE IN DE                       G   1781
09DD  D2 F0 09    JNC  X09F0 JMP IF HL ≥ DE - STRING IS BELOW STACK - STRING IN INSTRUCTION AREA  R  1782
09E0  2A E1 03    LHLD X03E1 LOAD HL WITH ADDR OF NXT-AVAILABLE BYTE IN STMT AREA (ALSO 2 BYTES  1783  BEFORE START OF SYMBOL & ARRAY TABLES
09E3  E7          RST  4   TEST PTRS AGAINST EACH OTHER                            1784
09E4  D2 F0 09    JNC  X09F0 JMP IF HL ≥ DE - IDENTITY GROUP IN STRING FORMULA TABLE  R  1785  IDENTITY GROUP IN VARIABLE REGION
09E7  3E 01       POP D MVI A,H'01' DUMMY PUT ADDR OF CHAR COUNT BYTE IN DE - FORCES REMOVAL OF IDENTITY GROUP  Q  1786  FROM STRING FORMULA TABLE IN CALL 1396
09E9  CD 96 13    CALL X1396 REMOVE 3 BYTE IDENTITY GROUP FROM FORMULA TABLE (IF MOST RECENT ENTRY)  M  1787  SPECIAL CASE ie A$ = B$
09EC  EB          XCHG  PUT ADDR OF CHAR COUNT BYTE IN HL (SAME AS ADDR FROM 09DC or 09EB)  1788  STRING FORMULA TABLE NOT INVOLVED
09ED  CD CA 11    CALL X11CA PUT IDENTITY GROUP OF STRING ON STRING STACK (STRING FORMULA TABLE)  MZ  1789  YET
09F0  CD 96 13  X09F0 CALL X1396 REMOVE IDENTITY GROUP FROM STRING FORMULA TABLE   M   1790
09F3  E3          XTHL PUT PTR TO VARIABLE VALUE REGION IN HL, PUT AN ADDR ON STACK TO BALANCE POP D AT 09E7  1791
09F4  CD 66 1B  X09F4 CALL X1B66 MOVE VALUE FROM HOLDING AREA TO AREA IN SYMBOL TABLE  M  1792
09F7  D1          POP  D   REMOVE 1 ADDR FROM STACK (FOR - DE IS PTR TO VAR VALUE REGION)  G  1793
09F8  E1          POP  H   FETCH PTR TO INSTRUCTION FROM STACK                     1794
09F9  C9          RET      END OF LET PROCESSING                                   I   1795
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
09FA  CD 81 14    CALL X1481 EVALUATE ARGUMENT TO A 1 BYTE VALUE (STOP AT DELIMITER)  M  1796  ON
09FD  7E          MOV  A,M FETCH NXT CHAR FROM INSTRUCTION STRING  (1 BYTE RETURNED IN REG E)  1797
09FE  47          MOV  B,A SAVE CHAR IN REG B ('GOTO' or 'GOSUB' TOKEN IN B)      G   1798
09FF  FE 8C       CPI  H'8C' IF CHAR IS A GOSUB TOKEN, JMP TO 0A07                1799
```



```
0A01  CA 07 0A          JZ   X0A07  } IF CHAR IS A 'GOSUB' TOKEN, JMP TO 0A07                J    1800
0A04  CF                RST  1    } SYNTAX CHECK FOR 'GOTO' TOKEN                           O    1801
0A05  88                DB '88'~~A0C~~ B }                                                      1802
0A06  2B                DCX  H  BACK-UP PTR TO INSTRUCTION (RST 1 ADVANCES PTR 1 BYTE TOO FAR)   1803
0A07  4B              X0A07 MOV  C,E  PUT 1 BYTE OF EVALUATED ARGUMENT IN REG C               K    1804
0A08  0C              X0A08 DCR  C  DECREMENT ARGUMENT                                         1805
0A09  78                MOV  A,B  PUT 'GOTO' OR 'GOSUB' TOKEN IN ACC                            1806
0A0A  CA 09 07          JZ   X0709 IF ARGUMENT HAS BEEN DECREMENTED TO 00, EXECUTE ROUTINE  9 TO 1807 COMPUTE SUBR ADDR FROM TOKEN
0A0D  CD F4 08        X0A0D CALL X08F4 ASSEMBLE NXT GROUP OF DIGITS INTO A LINE # (SKIP OVER CHAR'S TIL A M      1808 DELIMITER FOUND IE ',')
0A10  FE 2C             CPI  A',' IF DELIMITER CHAR IS NOT A COMMA, THEN 'ON' INSTRUCTION IS  ,                1809
0A12  C0                RNZ     } TERMINATED WITHOUT EXECUTING A 'GOTO' OR 'GOSUB'            @                1810
0A13  C3 08 0A          JMP  X0A08 LOOP TO PROCESS 'ON' INSTRUCTION                          C    1811
------------------------------------------------------------------------------------------------------------------------------------------------
0A16  CD 52 0C          CALL XOC52 EVALUATE EXPRESSION TO TRUE/FALSE VALUE (TRUE=-1, FALSE=0)  MR   1812
0A19  7E                MOV  A,M  } TEST NEXT CHAR IN INSTRUCTION STRING                        1813   IF
0A1A  FE 2C             CPI  A',' }                                                  ,          1814
0A1C  CC 01 07          CZ   X0701 IF CHAR IS A ',', USE RST 2 TO SCAN TO NEXT NON-SPACE CHAR (ALLOWS  LQ COMMA 1815 BEFORE 'THEN')
0A1F  FE 88             CPI  H'88' } IF CHAR IS A 'GOTO' TOKEN, JMP TO 0A27                     1816
0A21  CA 27 0A          JZ   X0A27 }                                                 J'         1817
0A24  CF                RST  1   } OTHERWISE, CHAR MUST BE A 'THEN' TOKEN            O          1818
0A25  AB              DB 'AB' ~~XRA E~~ } PERFORM SYNTAX CHECK                       +          1819
0A26  2B                DCX  H  BACK-UP PTR TO INSTRUCTION (RST 1 ADVANCES PTR 1 BYTE TOO FAR)  +    1820
0A27  E5              X0A27 PUSH H  SAVE PTR TO STMT ON STACK                                  1821
0A28  CD 25 18          CALL X1825 PERFORM SIGN TEST ON VALUE IN HOLDING AREA                M%   1822
0A2B  E1                POP  H  FETCH PTR TO STMT FROM STACK                                    1823
0A2C  CA 36 0A          JZ   X0A36 IF ARG TO 'IF' INSTRUCTION IS FALSE, JMP TO 0A36          J6   1824
0A2F  D7              X0A2F RST  2  SCAN TO NEXT NON-SPACE CHAR                              W    1825
0A30  DA 5A 09          JC   X095A IF CHAR IS A DIGIT, TREAT IT AS A GOTO, + DIGIT IS 1ST CHAR OF A LINE Z Z #  1826
0A33  C3 38 07          JMP  X0738 OTHERWISE, JMP TO COMMAND PROCESSOR TO BRANCH TO ROUTINE TO EXECUTE C8   1827 NEXT INSTRUCTION
0A36  16 01           X0A36 MVI  D,H'01' SET 'IF' CTR TO 1, ALLOWS COMPLEX IF THEN (IF THEN ELSE) ELSE STMTS 1828
0A38  CD 8C 09        X0A38 CALL X098C SCAN INSTRUCTION TIL NULL OR COLON FOUND, IF COLON FOUND AFTER 'IF' M TOKEN 1829, ADD 1 TO # IN D
0A3B  B7                ORA  A  } IF NULL WAS FOUND, THEN NO MATCHING ELSE, RETURN TO CNTL-C  7 CHECK 1830 + NEXT STMT IN PROGRAM
0A3C  C8                RZ      }                                                            H    1831
0A3D  D7                RST  2  SCAN TO NEXT NON-SPACE CHAR                                  W    1832
0A3E  FE 90             CPI  H'90' } IF CHAR AFTER COLON, IS NOT AN 'ELSE', SCAN INSTRUCTION FOR NEXT  1833 COLON OR A NULL
0A40  C2 38 0A          JNZ  X0A38 }                                                         88   1834
0A43  15                DCR  D  DECREMENT 'IF' CTR                                           1835
0A44  C2 38 0A          JNZ  X0A38 IF 'IF' CTR IS NOT ZERO, THEN INSTRUCTION MUST BE FURTHER SCANNED  88  1836 TO FIND A MATCHING 'ELSE', IF ONE EXISTS
0A47  C3 2F 0A          JMP  X0A2F MATCHING 'ELSE' WAS FOUND, SCAN TO NEXT INSTRUCTION & EXECUTE THAT  C/  1837 INSTRUCTION OR INSTRUCTION(S)
------------------------------------------------------------------------------------------------------------------------------------------------
0A4A  2B              X0A4A DCX  H  BACK-UP PTR TO INSTRUCTION                                +    1838   PART OF PRINT
0A4B  D7                RST  2  SCAN TO NEXT CHAR                                            W    1839
------------------------------------------------------------------------------------------------------------------------------------------------
0A4C  CA 98 0A          JZ   X0A98 IF COLON OR NULL AFTER PRINT, DO A CR & LF                J    1840
0A4F  C8              X0A4F RZ   IF NULL OR COLON LAST CHAR SCANNED, END OF PRINT EXECUTION   W    1841   PRINT
0A50  FE AA             CPI  H'AA' } IF CHAR IS 'USING' TOKEN, DO FORMATTED OUTPUT           *    1842
0A52  CA 5E 15          JZ   X155E }                                                         J^   1843
0A55  FE A6             CPI  H'A6' } IF CHAR IS 'TAB(' TOKEN, DO TAB FUNCTION                &    1844
0A57  CA C5 0A          JZ   X0AC5 }                                                         JE   1845
0A5A  FE A9             CPI  H'A9' } IF CHAR IS 'SPC(' TOKEN, DO SPACE FUNCTION              )    1846
0A5C  CA C4 0A          JZ   X0AC4 }                                                         JD   1847
0A5F  E5                PUSH H  SAVE PTR TO INSTRUCTION STRING                                    1848
0A60  FE 2C             CPI  A',' } IF CHAR IS ',' TOKEN, DO A 14 CHAR UNFORMATTED FIELD     ,    1849
0A62  CA 90 0A          JZ   X0A90 }                                                         JO   1850
0A65  FE 3B             CPI  A';' } IF CHAR IS ';', RESTORE SCAN PTR TO HL, SCAN TO NEXT CHAR ,   1851 CONTINUE PROCESSING
0A67  CA E6 0A          JZ   X0AE6 }                                                         J    1852
0A6A  C1                POP  B  REMOVE PTR TO INSTRUCTION FROM STACK INTO BC (JUST REMOVES ADDR  A  1853 FROM STACK)
0A6B  CD 52 0C          CALL XOC52 EVALUATE STRING (UP TO DELIMITER) TO VALUE IN HOLDING AREA  MR   1854
0A6E  E5                PUSH H  SAVE PTR TO INSTRUCTION ON STACK                                 1855
0A6F  F7                RST  6  TEST FOR VARIABLE TYPE                                           1856
0A70  CA 8C 0A          JZ   X0A8C IF CHAR STRING (OR STRING VARIABLE), PRINT CHAR STRING    J    1857
0A73  CD 81 21          CALL X2181 CONVERT # IN HOLDING AREA TO EQUIVALENT TO EQUIVALENT CHAR  M STRING 1858
0A76  CC FC 11          CALL X11FC GENERATE IDENTITY GROUP FOR STRING, & PUT GROUP IN STRING FORMULA  M TABLE 1859
```

```
0A79  2A 12 04          LHLD  X0412 PUT ADDR OF CHAR COUNT BYTE OF IDENTITY GROUP IN HL   *   1860
0A7C  3A 27 00          LDA   X0027 FETCH PRINT POSITION CTR                            :'  1861
0A7F  86                ADD   M     ADD LENGTH OF STRING FOR CONVERTED #                    1862
0A80  FE 48             CPI   A'H'  } IF PRINT POSITION IS AT END OF LINE, PRINT CRLF + NULLS 1863
0A82  D4 98 0A          CNC   X0A98 }                                               T   1864
0A85  CD 44 12          CALL  X1244 PRINT CHAR STRING OF CONVERTED # #              MD  1865
0A88  3E 20             MVI   A,A' ' } PRINT A SPACE AFTER #                        >   1866
0A8A  DF                RST   3     }                                               _   1867
0A8B  B7                ORA   A     RESET ZERO BIT (ACC CONTAINS A 'SP')            7   1868
0A8C  CC 44 12    X0A8C CZ    X1244 PRINT CHAR STRING                               LD  1869
0A8F  E1                POP   H     FETCH PTR TO INSTRUCTION FROM STACK                 1870
0A90  C3 4A 0A          JMP   X0A4A JMP TO CONTINUE SCANNING & PROCESSING PRINT INSTR CJ  1871
0A93  3E 00       X0A93 MVI   M,H'0D' PUT NULL AT END OF LINE BUFR                    6   1872
0A95  21 59 03          LXI   H,X0359 RESET HL TO POINT TO 1 LESS THAN START OF LINE BUFR !Y 1873
0A98  3E 00       X0A98 MVI   A,H'00' }                                               >   1874   PRINT CR LF & NULLS
0A9A  32 27 00          STA   X0027 } PRINT CR                                      2'  1875
0A9D  DF                RST   3     }                                               _   1876   CLEAR PRINT POSITION CTR
0A9E  3E 0A             MVI   A,H'0A' } PRINT LF                                    >   1877
0AA0  DF                RST   3     }                                               _   1878
0AA1  3A 26 00    X0AA1 LDA   X0026 FETCH NULL CTR INTO ACC                         :&  1879
0AA4  3D          X0AA4 DCR   A     DECREMENT NULL CTR                              =   1880
0AA5  32 27 00          STA   X0027 - EVENTUALLY PRINT POSITION CTR WILL BE ZEROED  2'WHEN 1881 EXITING LOOP
0AA8  C8                RZ          RETURN TO CNTL C CHECK OR ROUTINE THAT CALLED ØA98 H   1882
0AA9  F5                PUSH  PSW   STORE ACC (NULL CTR STORED ON STACK)    "ØAA1       1883
0AAA  AF                XRA   A     PUT A NULL IN ACC                               /   1884
0AAB  DF                RST   3     PRINT NULL                                      _   1885
0AAC  F1                POP   PSW   RESTORE ACC (FETCH NULL CTR FROM STACK)             1886
0AAD  C3 A4 0A          JMP   X0AA4 LOOP UNTIL ALL NULLS HAVE BEEN PRINTED          CS  1887
0AB0  3A 27 00    X0AB0 LDA   X0027 FETCH OUTPUT CHAR COUNT                          :'  1888   14 CHAR UNFORMATTED FIELD
0AB3  FE 38             CPI   A'8' } IF AT PRINT POSITION 56 or LARGER, PRINT CRLF & NULLS 8   1889
0AB5  D4 98 0A          CNC   X0A98 }                                               T   1890
0AB8  D2 E6 0A          JNC   X0AE6 RESTORE SCAN PTR TO HL, SCAN TO NXT CHAR, CONTINUE PROCESSING 1891 AFTER DOING CRLF NULLS
0ABB  D6 0E       X0ABB SUI   H'0E' } SUBTRACT 14. FROM FIELD LENGTH, REPEAT UNTIL RESULT  V   1892
0ABD  D2 BB 0A          JNC   X0ABB } IS MINUS.                                     R:  1893
0AC0  2F                CMA   COMPLEMENT REMAINDER, # IN ACC IS # OF SPACES-1 TO PRINT / TO 1894 GET TO MULTIPLE OF 14 POSITION
0AC1  C3 DD 0A          JMP   X0ADD PRINT SPACES                                    CJ  1895
0AC4  37          X0AC4 STC     CARRY SET FOR SPACE FUNCTION                        7   1896   SPACE FUNCTION FOR PRINT
0AC5  F5          X0AC5 PUSH PSW CARRY RESET FOR TAB FUNCTION, SAVE CARRY STATUS & TOKEN ON STACK 1897
0AC6  CD 80 14          CALL  X1480 FETCH # OF BLANKS TO PRINT FROM TAB OR SPACE ARGUMENT M   1898   TAB FUNCTION FOR PRINT
0AC9  CF                RST   1     } SYNTAX CHECK FOR ')' AFTER ARGUMENT            O   1899
0ACA  29          DB ')'~~0A0~~H    }                                               )   1900
0ACB  2B                DCX   H     BACK UP PTR TO INSTRUCTION                      +   1901
0ACC  F1                POP   PSW   FETCH TOKEN FROM STACK                              1902
0ACD  FE A9             CPI   H'A9' TEST FOR SPC( TOKEN                             )   1903
0ACF  E5                PUSH  H     SAVE PTR TO INSTRUCTION ON STACK                    1904
0AD0  3E FF             MVI   A,H'FF' PUT -1 IN ACC FOR SPC FUNCTION                >   1905
0AD2  CA D9 0A          JZ    X0AD9 JMP IF SPC FUNCTION, ADD ARGUMENT (# of BLANKS) TO -1 JY 1906
0AD5  3A 27 00          LDA   X0027 FETCH SCAN PTR                                  :'  1907
0AD8  2F                CMA   COMPLEMENT SCAN PTR TO DO SUBTRACT                    /   1908
0AD9  83          X0AD9 ADD   E ADD BLANK COUNT TO -1 (SPC) or NEGATED SCAN PTR (TAB)    1909
0ADA  D2 E6 0A          JNC   X0AE6 IF RESULT IS NEGATIVE, SKIP PRINTING OF ANY BLANKS R   1910
0ADD  3C          X0ADD INR   A     ADD 1 TO # OF BLANKS TO PRINT                   <   1911
0ADE  47                MOV   B,A   USE REG B TO STORE BLANK CTR                    G   1912
0ADF  3E 20             MVI   A,A' ' PUT 'SP' CHAR IN ACC                           >   1913
0AE1  DF          X0AE1 RST   3     PRINT 'SP'                                      _   1914
0AE2  05                DCR   B     DECREMENT SPACE CTR                             -   1915
0AE3  C2 E1 0A          JNZ   X0AE1 PRINT 'SP'S UNTIL CTR IS ZERO                   B   1916
0AE6  E1          X0AE6 POP   H     FETCH PTR TO INSTRUCTION FROM STACK                 1917
0AE7  D7                RST   2     SCAN TO NEXT CHAR                           W   1918
0AE8  C3 4F 0A          JMP   X0A4F JMP TO CONTINUE PROCESSING PRINT INSTRUCTION     CO  1919
```

```
0AE0  3F              X0AE0  CMC                                    ?   1920
0AE1  52                     MOV  D,D                               R   1921  CHAR STRING
0AE2  45                     MOV  B,L                               E   1922  "?REDO FROM START"
0AE3  44                     MOV  B,H                               D   1923
0AE4  4F                     MOV  C,A                               O   1924
0AE5  20                     DATA A' '                                  1925
0AE6  46                     MOV  B,M                               F   1926
0AE7  52                     MOV  D,D                               R   1927
0AE8  4F                     MOV  C,A                               O   1928
0AE9  4C                     MOV  C,L                               M   1929
0AEA  20                     DATA A' '                                  1930
0AEB  53                     MOV  D,E                               S   1931
0AEC  54                     MOV  D,H                               T   1932
0AED  41                     MOV  B,C                               A   1933
0AEE  52                     MOV  D,D                               R   1934
0AEF  04 00 0A               CNC  X0A00                             T   1935
0AF2  00                     NOP                                        1936  ERROR CAUSED BY INPUT OR READ
0AF3  3A 02 03       X0AF3   LDA  X0302 } TEST BRANCHING FLAG       :R  1937  VALUE NOT MATCHING VARIABLE TYPE
0AF6  B7                     ORA  A     } INPUT ØØ  READ AF         7   1938  OR ILLEGAL CHAR TYPED
0AF7  C2 A3 04               JNZ  X04A3 IF 'READ' STMT, THEN PRINT "SYNTAX ERROR", USE LINE # OF  B#DATA 1939  STMT (COMMA WAS NOT WHERE EXPECTED)
0AFA  C1                     POP  B REMOVE 1 ADDR FROM STACK        A   1940
0AFB  21 EB 0A               LXI  H,X0AEB LOAD HL WITH ADDR OF CHAR STRING "?REDO FROM START"  !   1941
0AFE  CD 41 12               CALL X1241 PRINT CHAR STRING           MA  1942
0B01  C3 E2 05               JMP  X05E2 RESTART INPUT INSTRUCTION   C   1943
0B04  FE 22                  CPI  A'"' TEST 1ST NON-SPACE CHAR AFTER INPUT TOKEN  "  1944
0B06  3E 00                  MVI  A,H'00' } CLEAR PRINT SUPPRESSION FLAG  >  1945  INPUT
0B08  32 A2 03               STA  X03A2   }                         2"  1946
0B0B  C2 1A 0B               JNZ  X0B1A JMP IF CHAR IS NOT A '"'    B#  1947
0B0E  CD F0 11               CALL X11F0 GENERATE CHAR COUNT FOR STRING WITHIN QUOTES, CHAR COUNT IS IN C  M  1948  PRINT MESSAGE WITHIN QUOTES
0B11  CF                     RST  1 } SYNTAX CHECK FOR ';' AFTER CHAR STRING WITHIN QUOTES  O  1949
0B12  3B             DB';'~~BC~~ ~~SP~~ }                            ;   1950
0B13  E5             X0B13   PUSH H  SAVE PTR TO INSTRUCTION ON STACK                1951
0B14  CD 44 12               CALL X1244 PRINT CHAR STRING WITHIN QUOTES  MD  1952
0B17  E1                     POP  H  FETCH PTR TO INSTRUCTION FROM STACK    1953
0B18  E5             X0B18   PUSH H  SAVE PTR TO INSTRUCTION ON STACK       1954
0B19  CD 03 11               CALL X1103 TEST IF INPUT IS BEING USED AS A DIRECT COMMAND - ILLEGAL  M3-  1955  PRINT ERROR MESSAGE-E
0B1C  CD E6 05               CALL X05E6 PRINT "?␣", INPUT CHAR STRING INTO LINE BUFR, LAST CHAR IS A  MNULL  1956
0B1F  23                     INX  H  ADVANCE PTR TO POINT TO 1ST CHAR IN LINE BUFR  #  1957
0B20  7E                     MOV  A,M } TEST FIRST CHAR IN LINE BUFR        1958
0B21  B7                     ORA  A   }                             7   1959
0B22  2B                     DCX  H  BACK-UP PTR TO 1 BYTE BEFORE 1ST CHAR IN LINE BUFR  +  1960
0B23  C1                     POP  B  REMOVE 1 ADDR FROM STACK       A   1961
0B24  CA F8 07               JZ   X07F8 IF ONLY A CR WAS TYPED IN RESPONSE TO THE '?␣', EXECUTE 'END',  J  1962  CONTINUE CAN BE USED TO RESTART PROGRAM AT
0B27  C5                     PUSH B  PUT ADDR BACK ON STACK (PTR TO INSTRUCTION)  E  1963
0B28  C3 30 0B               JMP  X0B30                             C:  1964
0B2B  E5             X0B2B   PUSH H  PUT PTR TO INSTRUCTION ON STACK        1965
0B2C  2A E7 03               LHLD X03E7 LOAD HL WITH ADDR IN PTR USED TO SCAN 'DATA' STMTS  *  1966  READ
0B2F  F6 AF                  ORI  H'AF' Dummy XRA A } 'INPUT' STORES ØØ AT Ø3D2  /  1967
0B31  32 02 03               STA  X0302             } 'READ' STORES AF AT Ø3D2  2R  1968
0B34  E3                     XTHL PUT PTR TO INSTRUCTION IN HL, PUT PTR TO STRING OF CHAR'S FOR VALUE ON STACK  1969
0B35  01 CF 2C               ~~LXI B,X2CCF~~ Dummy RST1 DB ',' SYNTAX CHECK FOR COMMA  O,  1970
0B38  CD E0 0E               CALL X0EE0 FIND VARIABLE OR ARRAY ELEMENT IN THE APPROPRIATE TABLE  M  1971
0B3B  E3                     XTHL PUT PTR TO STRING OF CHAR'S FOR VALUE IN HL, PUT PTR TO INSTRUCTION ON STACK  1972
0B3C  D5                     PUSH D PUT PTR TO VARIABLE VALUE REGION ON STACK  U  1973
0B3D  7E                     MOV  A,M } IS CHAR POINTED TO BY HL A COMMA ? (COULD BE CHAR BEFORE START OF CHAR BUFR)  ~  1974
0B3E  FE 2C                  CPI  A',' } (THIS IS THE CASE FOR HL POINTING TO 'DATA' STMT FOR 'READ', OR #'S SEPARATED BY COMMA'S IN LINE BUFR)  ,  1975
0B40  CA 58 0B               JZ   X0B58 YES, CHAR IN DATA STMT IS A COMMA, JMP  JC  1976
0B43  3A 02 03               LDA  X0302 } TEST BRANCHING FLAG       :R  1977
0B46  B7                     ORA  A     } INPUT ØØ  READ AF         7   1978
0B47  C2 B5 0B               JNZ  X0BB5 JMP IF 'READ' INSTRUCTION BEING PROCESSED - FIND A 'DATA'  B5  1979
                                                                          STMT IN STMT AREA
```

```
0355  3F 3F          MVI  A,A'?'  } PRINT '?' AS PROMPT CHAR          >?   1980
0357  DF             RST  3                                           N    1981
0358  CD E6 05       CALL X05E6   PRINT '?␣', INPUT CHAR STRING INTO LINE BUFR, LAST CHAR IS A NULL  M  1982
035B  F7       X035B   RST  6     TEST VARIABLE TYPE (Loc 03A4)                                    1983  EVALUATE CHAR STRING TO VALUE
035C  F5               PUSH PSW   SAVE (TYPE-3) & FLAGS ON STACK                                    1984  ARITHMETIC OR STRING TYPE
035D  C2 7A 08         JNZ  X087A JMP IF VARIABLE IS ARITHMETIC TYPE                          B     1985
0360  D7               RST  2 SCAN TO CHAR (NON-SPACE) AFTER DELIMITER, CHAR IN ACC            W     1986  STRING VARIABLE
0361  57               MOV  D,A } IF 1ST CHAR IS '"', USE THIS AS THE DELIMITER CHAR           W     1987
0362  47               MOV  B,A } IN 1200 TO FIND THE END OF THE STRING                        G     1988
0363  FE 22            CPI  A'"'                                                                "    1989
0365  CA 60 08         JZ   X0860                                                              J    1990
0368  16 3A            MVI  D,A':' } IF 1ST CHAR IS NOT '"', USE ',' & ':' AS DELIMITERS TO    :    1991
036A  06 2C            MVI  B,A',' } FIND END OF STRING IN 1200                                ,    1992
036C  2B               DCX  H  BACK-UP PTR TO STRING - ALLOWS 1ST CHAR TO BE TREATED AS PART OF +    1993  STRING
036D  CC 00 12   X036D CALL X1200 GENERATE A CHAR COUNT FOR IDENTITY GROUP OF STRING VALUE     M    1994
0370  F1       X0370   POP  PSW  PUT (TYPE-3) OF VARIABLE IN ACC                                  1995
0371  EB               XCHG PUT PTR TO DATA STREAM IN DE                                        1996
0372  21 81 08         LXI  H,X0881 } PUT PTR TO VARIABLE VALUE REGION IN HL                !   1997
0375  E3               XTHL         } PUT ADDR 0881 ON STACK, IMPLIED JMP TO LOOP UNTIL ALL VAR'S  1998  SEPARATED BY COMMA'S HAVE BEEN PROCESSED
0376  D5               PUSH D PUT PTR TO DATA STREAM ON STACK                                U    1999  RETURN TO 0881
0377  C3 90 09         JMP  X0990 PUT NUMERIC VALUE, STORE IN VARIABLE VALUE REGION AFTER    C=   2000  APPROPRIATE TYPE CONVERSION, IF NECESSARY
037A  D7       X037A   RST  2 SCAN TO FIRST CHAR OF STRING TO BE CONVERTED TO A VALUE         W    2001  ARITHMETIC VARIABLE
037B  CD 43 20         CALL X2043 CONVERT STRING TO NUMERIC VALUE                             MC   2002
037E  C3 70 08         JMP  X0870 STORE VALUE IN VARIABLE VALUE REGION, CONTINUE TO PROCESS C 'READ' 2003  OR 'INPUT' STMT
0381  2B       X0381   DCX  H  BACK-UP PTR TO DATA STREAM (IN 'DATA' STMT OR LINE BUFR)       +    2004
0382  D7               RST  2 SCAN TO NEXT NON-SPACE CHAR                                     W    2005
0383  CA 88 08         JZ   X0888 JMP IF NULL OR COLON - END OF INPUT STREAM, CONTINUE TO     J    2006  PROCESS 'READ' OR 'INPUT' TIL ALL VARIABLES HAVE BEEN SATISFIED
0386  FE 2C            CPI  A',' } IF CHAR IS NOT A COMMA, 'READ' -> "SYNTAX ERROR" MESSAGE  ,    2007
0388  C2 FE 0A         JNZ  X0AFE} ILLEGAL CHAR IN INPUT STREAM   'INPUT' -> RESTART BASIC PROGRAM AFTER 2008  MESSAGE "?REDO␣FROM␣START"
0388  E3       X0388   XTHL PUT PTR TO INSTRUCTION IN HL, PUT PTR TO DATA STREAM ON STACK      $    2009
038C  2B               DCX  H BACK-UP PTR TO INSTRUCTION                                      +    2010
038D  D7       ANOTHER VARIABLE  RST  2 SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION (NULL, COLON, OR ANOTHER 2011  VARIABLE NAME)
038E  C2 41 03 <ANOTHER VARIABLE JNZ  X0341 JMP TO PROCESS ANOTHER VARIABLE                   BA   2012
0391  D1               POP  D PUT PTR TO INPUT STREAM IN DE                                   Q    2013  READ OR INPUT ALMOST DONE
0392  3A 02 03         LDA  X0302 } TEST BRANCHING FLAG                                       :R   2014
0395  B7               ORA  A     } INPUT 00  READ FF                                         7    2015
0396  EB       END OF 'READ' XCHG PUT PTR TO INPUT STREAM IN HL, PUT PTR TO INSTRUCTION IN DE      2016
0397  C2 E6 07 <END OF 'READ' JNZ  X07E6 IF 'READ' INSTR, SAVE PTR TO 'DATA' IN 03E7 FLAG    B    2017
039A  36               ORA  M TEST IF LAST CHAR IN DATA STREAM (LINE BUFR) IS A NULL - IF NOT 6,  2018  PRINT MESSAGE IN 0B9F
039B  21 A4 0B         LXI  H,X0BA4 LOAD HL WITH ADDR OF CHAR STRING "?EXTRA␣IGNORED"         !$   2019
039E  D5               PUSH D PUT PTR TO INSTRUCTION ON STACK                                 U    2020
039F  C4 41 12         CNZ  X1241 PRINT CHAR STRING                                           DA   2021
03A2  E1               POP  H PUT PTR TO INSTRUCTION IN HL                                         2022
03A3  C9               RET  END OF PROCESSING 'INPUT' INSTRUCTION                             I    2023
03A4  3F       X0BA4   CMC                                                                    ?    2024  CHAR STRING
03A5  45               MOV  B,L                                                               E    2025  "?EXTRA␣IGNORED"
03A6  58               MOV  E,B                                                               X    2026
03A7  54               MOV  D,H                                                               T    2027
03A8  52               MOV  D,D                                                               R    2028
03A9  41               MOV  B,C                                                               A    2029
03AA  20               DATA A' '                                                                   2030
03AB  49               MOV  C,C                                                               I    2031
03AC  47               MOV  B,A                                                               G    2032
03AD  4E               MOV  C,M                                                               N    2033
03AE  4F               MOV  C,A                                                               O    2034
03AF  52               MOV  D,D                                                               R    2035
03B0  45               MOV  B,L                                                               E    2036
03B1  C4 0D 0A         CNZ  X0A0D                                                             D    2037
03B4  00               NOP                                                                         2038
03B5  CD 8C 09   X03B5 CALL X098C SCAN STMT POINTED TO BY HL TIL NULL FOUND                   M    2039  'READ' - FIND A 'DATA' STMT IN STMT AREA
```

```
0B88  B7          ORA  A   TEST CHAR
0B89  C2 CE 0B    JNZ  X0BCE IF CHAR IS A ':', THEN JMP TO TEST IF DATA INSTRUCTION, IF CHAR IS A NULL, LOOK AT NXT STMT
0B8C  23          INX  H   } TEST NXT 2 BYTES IN STMT. AREA (CHAIN ADDR)
0B8D  7E          MOV  A,M }
0B8E  23          INX  H   }
0B8F  B6          ORA  M   }
0B90  1E 04       MVI  E,H'04' } IF CHAIN ADDR IS ØØØØ, THEN NOT ENOUGH DATA STMTS TO
0B92  CA B7 04    JZ   X04B7  } SATISFY ALL OF THE READ STMT'S, PRINT ERROR MESSAGE 'OUT OF DATA'
0B95  23          INX  H   } PUT LINE # OF STMT IN DE
0B96  5E          MOV  E,M }
0B97  23          INX  H   }
0B98  56          MOV  D,M }
0B99  EB          XCHG       } STORE LINE # AT Ø3CF IN FLAG AREA (USED FOR ERROR
0B9A  22 CF 03    SHLD X03CF } MESSAGE, IF ERROR OCCURS
0B9D  EB          XCHG       }
0B9E  D7    X0BCE RST  2     SCAN TO NXT NON-SPACE CHAR IN STMT
0B9F  FE 83       CPI  H'83' } TEST CHAR, IF NOT A 'DATA' TOKEN, SCAN OVER INSTRUCTION TO
0BA1  C2 25 0B    JNZ  X0B85 } THE NEXT COLON
0BA4  C3 50 0B    JMP  X0B53 IF CHAR IS 'DATA' TOKEN, THEN USE NXT CHAR'S AS THE STRING TO BE CONVERTED TO A VALUE FOR 'READ'
0BA7  11 00 00    LXI  D,X0000 LOAD DE WITH ØØØØ, FORCES EXIT FROM Ø457 UPON FINDING MOST RECENT 'FOR' TOKEN ON THE STACK
0BAA  C4 E0 0E    CNZ  X0EE0 IF NON-SPACE CHAR AFTER 'NEXT' TOKEN NOT A NULL OR ':', FIND VAR IN SYMBOL TABLE
0BAD  22 03 03    SHLD X0303 STORE PTR TO STMT IN Ø303 (TEMP STORAGE IN FLAG AREA)
0BB0  CD 57 04    CALL X0457 TEST STACK CONTENTS TO SEE IF A 'FOR' GROUP OF BYTES IS ACTIVE & NEAR CURRENT SP POSITION
0BB3  C2 AF 04    JNZ  X04AF IF 'FOR' TOKEN NOT FOUND AT SP+4, PRINT ERROR MESSAGE "NEXT WITHOUT FOR"
0BB6  F9          SPHL ADJUST STACK POINTER TO POINT TO SIGN-OF-STEP BYTE
0BB7  D5          PUSH D PUT PTR TO VARIABLE (FROM FOR-GROUP OF BYTES ON STACK) ON STACK - NOT MATCHED BY POP - ADJUSTS STACK PTR FOR ØC41
0BB8  7E          MOV  A,M FETCH SIGN-OF-STEP BYTE FROM FOR-GROUP
0BB9  23          INX  H   ADVANCE PTR TO FOR-GROUP OF BYTES
0BBA  F5          PUSH PSW PUT SIGN-OF-STEP BYTE ON STACK
0BBB  D5          PUSH D SAVE PTR TO VARIABLE (FROM FOR-GROUP OF BYTES ON STACK) ON STACK
0BBC  7E          MOV  A,M FETCH [(TYPE-3) OF STEP] BYTE FROM FOR GROUP
0BBD  23          INX  H ADVANCE PTR TO FOR-GROUP OF BYTES
0BBE  D7          ORA  A TEST TYPE OF STEP (INTEGER IS -1 IN ACC, SNG-PREC IS 1 IN ACC)
0BBF  FA 09 0C    JM   X0C09 JMP IF USING INTEGER ARITHMETIC
0BC2  CD 43 18    CALL X1843 FETCH STEP SIZE FROM FOR-GROUP INTO HOLDING AREA
0BC5  E3          XTHL SAVE PTR TO FOR-GROUP ON STACK, PUT PTR TO VARIABLE VALUE REGION IN HL
0BC6  E5          PUSH H PUT PTR TO VARIABLE BACK ON STACK
0BC7  CD 90 18    CALL X1890 FETCH VAR VALUE FROM SYMBOL TABLE, ADD STEP TO VARIABLE VALUE
0BCA  E1          POP  H FETCH PTR TO VARIABLE FROM STACK
0BCB  CD 50 18    CALL X1850 STORE THE RESULT OF THE ADDITION BACK INTO THE VARIABLE VALUE REGION IN SYMBOL TABLE
0BCE  E1          POP  H FETCH PTR TO FOR-GROUP FROM STACK
0BCF  CD 54 18    CALL X1854 FETCH LIMIT VALUE FROM FOR-GROUP INTO BCDE
0BD2  E5          PUSH H SAVE PTR TO FOR-GROUP ON STACK
0BD3  CD A1 18    CALL X18A1 PERFORM SNG-PREC MAGNITUDE TEST, TEST IF SUM EXCEEDS LIMIT
0BD6  C3 32 0C    JMP  X0C32
0BD9  23    X0C09 INX  H } ADVANCE PTR TO FOR-GROUP PAST 4 DUMMY BYTES PUT ON
0BDA  23          INX  H } STACK AT Ø76E & Ø76F - REMEMBER THE 17 BYTE ALIGNMENT
0BDB  23          INX  H }
0BDC  23          INX  H }
0BDD  4E          MOV  C,M } FETCH 2 BYTE INTEGER FORMAT STEP SIZE INTO BC
0BDE  23          INX  H   }
0BDF  46          MOV  B,M }
0BE0  23          INX  H   }
0BE1  E3          XTHL PUT PTR TO VARIABLE VALUE REGION IN HL, PUT PTR TO FOR-GROUP ON STACK
0BE2  5E          MOV  E,M } FETCH VARIABLE VALUE FROM SYMBOL TABLE INTO DE
0BE3  23          INX  H   }
0BE4  56          MOV  D,M }
0BE5  E5          PUSH H SAVE PTR TO VARIABLE VALUE REGION (HI-BYTE)
0BE6  69          MOV  L,C } PUT STEP SIZE IN HL
0BE7  60          MOV  H,B }
```

NEXT

```
0C18  CD 55 10          CALL  X1055 PERFORM INTEGER ADDITION, ADD STEP SIZE TO VARIABLE VALUE          MU   2100
0C1B  3A A4 03          LDA   X03A4 } FETCH & TEST OPERATION-TYPE BYTE FROM FLAG AREA                 :$   2101
0C1E  FE 04             CPI   H'04' }                                                                      2102
0C20  CA 3E 19          JZ    X193E IF INTEGER ADD OVERFLOWED, RESULT IN HOLDING AREA WOULD BE SNG-PREC>TYPE 2103  PRINT ERROR MESSAGE "OVERFLOW"
0C23  EB                XCHG  PUT RESULT OF ADDITION IN DE (1055 PUTS RESULT IN HL + HOLDING AREA)        2104
0C24  E1                POP   H FETCH PTR TO VARIABLE VALUE REGION FROM STACK                            2105
0C25  72                MOV   M,D } STORE THE RESULT OF THE ADDITION BACK INTO THE VARIABLE               2106
0C26  2D                DCX   H   } VALUE REGION IN SYMBOL TABLE                                    +    2107
0C27  73                MOV   M,E }                                                                      2108
0C28  E1                POP   H FETCH PTR TO FOR-GROUP FROM STACK                                        2109
0C29  D5                PUSH  D PUT SUM ON STACK                                                U    2110
0C2A  5E                MOV   E,M } FETCH LIMIT VALUE FROM FOR-GROUP INTO DE                         ^    2111
0C2B  23                INX   H   }                                                                 #    2112
0C2C  56                MOV   D,M }                                                                 V    2113
0C2D  23                INX   H   }                                                                 #    2114
0C2E  E3                XTHL  PUT SUM IN HL, PUT PTR TO FOR-GROUP ON STACK                               2115
0C2F  CD CC 1B          CALL  X1BCC PERFORM INTEGER MAGNITUDE TEST, TEST IF SUM EXCEEDS LIMIT           ML   2116
0C32  E1             >0C32  POP   H FETCH PTR TO FOR-GROUP FROM STACK                                    2117
0C33  C1                POP   B FETCH SIGN-OF-STEP BYTE FROM STACK                                 A    2118
0C34  90                SUB   B TEST IF RESULT OF MAGNITUDE TEST + SIGN-OF-STEP INDICATE END OF 'FOR'    2119  INSTRUCTION
0C35  CD 54 1B          CALL  X1B54 PUT 'FOR' LINE # IN DE, PUT 'FOR' TERMINATOR ADDRESS IN BC          MT   2120
0C38  CA 44 0C          JZ    X0C44 JMP IF 'FOR' INSTRUCTION IS FINISHED                                JD   2121
0C3B  EB                XCHG  } STORE 'FOR' LINE # IN FLAG AREA (03D7 IS LOC FOR CURRENT                  2122
0C3C  22 D7 03          SHLD  X03D7 }                INSTRUCTION LINE #)                           "W   2123
0C3F  69                MOV   L,C } PUT 'FOR' TERMINATOR ADDRESS IN HL (BYTE AFTER TERMINATOR IS 1ST BYTE 2124
0C40  60                MOV   H,B }                                  OF NXT INSTRUCTION AFTER 'FOR')     2125
0C41  C3 79 07          JMP   X0779 PUT 'FOR' TOKEN ON STACK, DO CNTL-C CHK, CONTINUE EXECUTING INSTR'S FOLLOWING 2126  'FOR' INSTRUCTION
0C44  F9             >0C44  SPHL  ELIMINATE 'FOR' GROUP FROM STACK BY MOVING STACK PTR                    2127
0C45  2A D3 03          LHLD  X03D3 FETCH PTR TO CURRENT INSTRUCTION FROM FLAG AREA (see 0BDC)          *S   2128
0C48  7E                MOV   A,M } TEST NXT CHAR IN 'NEXT' INSTRUCTION                                  2129
0C49  FE 2C             CPI   A',' }                                                               ,    2130
0C4B  C2 70 07          JNZ   X0770 IF CHAR IS NOT A ',', THEN 'NEXT INSTRUCTION IS COMPLETED, DO       B    2131  CNTL-C CHK
0C4E  D7                RST   2 SCAN TO NXT NON-SPACE CHAR IN INSTRUCTION                           W    2132
0C4F  CD DA 0B          CALL  X0BDA EXECUTE 'NEXT' FOR 2ND VARIABLE (NOTE: CALL PUTS ADDR ON STACK TO RETURN BALANCE USUAL CNTL-C CHK ADDR ON STACK FOR 0457  2133
0C52  2B             >0C52  DCX   H       DECREMENT PTR TO STMT                                  +    2134
0C53  16 00          >0C53  MVI   D,H'00' SET PRECEDENCE BYTE TO ZERO - INITIALIZE TO EVALUATE STRING      2135
0C55  D5             >0C55  PUSH  D  PUT PRECEDENCE BYTE OF PREVIOUS OPERATOR ON STACK.             U    2136
0C56  0E 01                 MVI   C,H'01' SET BYTE COUNT TO 1 FOR 0487 ROUTINE - TEST FOR ROOM FOR 2 BYTES 2137
0C58  CD 87 04              CALL  X0487 TEST FOR ENOUGH AVAILABLE MEMORY                            M    2138
0C5B  CD CB 0D              CALL  X0DCB TEST LEADING CHAR - TAKE APPROPRIATE ACTION TO EVALUATE PART OF NK STRING 2139
0C5E  22 D5 03              SHLD  X03D5 SAVE PTR TO STRING AT 03D5                                  "U   2140
0C61  2A D5 03       >0C61  LHLD  X03D5 FETCH PTR TO STRING FROM ISTORAGE AT 03D5                   *U   2141
0C64  C1             >0C64  POP   B    FETCH PRECEDENCE BYTE OF PREVIOUS OPERATOR FROM STACK         A    2142
0C65  7E                    MOV   A,M FETCH NEXT CHAR FROM STRING INTO ACC - SHOULD BE AN OPERATOR          2143
0C66  16 00                 MVI   D,H'00' SET OPERATOR REG TO 0 -OR- CLEAR UPPER BYTE OF PRECEDENCE REGISTER 2144  - see 0C91 & 0C9E
0C68  D6 37          >0C68  SUI   H'87' SUBTRACT 87 FROM CHAR - 1 BEYOND LARGEST TOKEN FOR 2-ARGUMENT (see 0C82-0C84) V7   2145  ARITHMETIC TOKEN
0C6A  DA 82 0C              JC    X0C82 JMP IF ARITHMETIC TOKEN OR ASCII DELIMITER                  Z    2146
0C6D  FE 03                 CPI   H'03' } JMP IF TOKEN IS 1-ARGUMENT FUNCTION TOKEN - SGN (BA) TO MID (B4)  2147
0C6F  D2 82 0C              JNC   X0C82 } OTHERWISE TOKEN IS > (B7), = (B8), < (B9)  ACC HAS 0 FOR >, 1 FOR =, 2 FOR < 2148  BIT WEIGHTS FOR RELATIONAL OPERATORS
0C72  FE 01                 CPI   H'01' SET CARRY FLAG IF '>' TOKEN                                  2149  > 01
0C74  17                    RAL         MULTIPLY OFFSET BY 2                                        2150  = 02
0C75  AA                    XRA   D     COMBINE WITH PREVIOUS RELATIONAL OPERATOR OFFSET            *    2151  < 04
0C76  BA                    CMP   D     TEST IF SAME OPERATOR USED TWICE                            :    2152
0C77  57                    MOV   D,A   SAVE OFFSET - OF ACCUMULATED OPERATORS                      W    2153
0C78  DA A9 04              JC    X04A9 PRINT ERROR MESSAGE "SYNTAX ERROR" - IF OPERATOR USED TWICE Z)   2154
0C7B  22 CD 03              SHLD  X03CD SAVE PTR TO LAST OPERATOR IN 03CD FLAG AREA                  "M   2155
0C7E  D7                    RST   2     SCAN TO NEXT NON-SPACE CHAR - COULD BE ANOTHER OPERATOR      W    2156
0C7F  C3 68 0C              JMP   X0C68 LOOP UNTIL NON-RELATIONAL OPERATOR OR CHAR FOUND             C    2157
0C82  7A             >0C82  MOV   A,D } TEST OPERATOR REGISTER                                         2158
0C83  B7                    ORA   A   }                                                             7    2159
```

```
0C84  C2 12 0D <----------  JNZ  X0D12 JMP IF EVALUATING RELATIONAL EXPRESSION (<, >, =>, etc.)   B     2160
0C87  7E                   MOV  A,M  PUT CHAR FROM STRING INTO ACC                               "     2161
0C88  22 C0 03             SHLD X03C0 SAVE PTR TO STRING IN FLAG AREA                            "M    2162
0C8B  D6 AE                SUI  H'AE' SUBTRACT AE FROM CHAR - SMALLEST TOKEN FOR ARITHMETIC TOKEN .    2163
0C8D  D8          <-DONE-  RC     RETURN IF CHAR IS SMALLER THAN A '+' TOKEN - DELIMITER  X IN  2164 STRING } ONLY 9 TWO-ARG ARITHMETIC
0C8E  FE 09                CPI  H'09' } TEST CHAR + RETURN IF CHAR IS LARGER THAN A '\' TOKEN - DELIMITER 2165 IN STRING } OPERATORS
0C90  D0          <-DONE-  RNC                                                                  P     2166
0C91  5F                   MOV  E,A  SAVE ARITHMETIC TOKEN-OFFSET IN REG E                       _     2167
0C92  3A A4 03             LDA  X03A4 } TEST TYPE OF COMPUTATION BEING USED TO EVALUATE THE STRING !$   2168
0C95  D6 03                SUI  H'03' }                                                          V     2169
0C97  B3                   ORA  E     } PERFORM TEST - JMP IF TYPE IS 03 + ARITHMETIC OPERATOR IS 3 +  2170
0C98  CA 2E 13 <----------  JZ   X132E } - JMP TO PERFORM STRING VARIABLE CONCATENATION          J.    2171
0C9B  21 71 00             LXI  H,X0071 LOAD HL WITH ADDR OF ARITHMETIC OPERATOR PRECEDENCE TABLE !     2172
0C9E  19                   DAD  D  COMPUTE ADDR OF PRECEDENCE BYTE FOR OPERATOR IN E              .     2173
0C9F  78                   MOV  A,B PUT PREVIOUS-OPERATOR PRECEDENCE INTO ACC                    .     2174
0CA0  56                   MOV  D,M PUT PRESENT-OPERATOR PRECEDENCE INTO REG D                   V     2175
0CA1  BA                   CMP  D  COMPARE THE TWO PRECEDENCE OPERATORS                          :     2176
0CA2  D0          <----------  RNC  IF CURRENT OPERATOR PRECEDENCE IS ≤ PREVIOUS OPERATOR, EXECUTE A RETURN (ACTUALLY AN IMPLIED JMP) TO FCN ROUTINE WHICH WILL COMBINE THE ARG 2177
0CA3  C5                   PUSH B  PUT PREVIOUS-OPERATOR PRECEDENCE ONTO STACK                   E     2178 ON THE STACK WITH THE ARG IN THE HOLDING
0CA4  01 61 0C             LXI  B,X0C61 } PUT LOOP ADDR ON STACK (IMPLIED JMP) - THIS IS THE     .     2179 AREA) - WHEN DONE - IMPLIED JMP TO 0C61
0CA7  C5                   PUSH B       } REASON FOR TEST AT 0C58                                E     2180
0CA8  7A                   MOV  A,D PUT PRESENT-OPERATOR PRECEDENCE IN ACC                       .     2181
0CA9  FE 7F                CPI  H'7F' } TEST PRESENT-OPERATOR PRECEDENCE + JMP IF '↑' OPERATION   .     2182
0CAB  CA F8 0C             JZ   X0CF8 }                                                          J     2183
0CAE  FE 51                CPI  A'Q'  } TEST PRESENT-OPERATOR PRECEDENCE + JMP IF 'AND' or 'OR' OPERATION Q  2184
0CB0  DA 05 0D             JC   X0D06 }                                                          Z     2185
0CB3  E6 FE                ANI  H'FE' MASK THE LSB OF THE PRESENT-OPERATOR PRECEDENCE BYTE       .     2186
0CB5  FE 7A                CPI  H'7A' } JMP IF 'MOD' or '\' OPERATION                            .     2187
0CB7  CA 06 0D   <----------JZ   X0D06 }                                                          J     2188
0CBA  21 12 04   X0CBA     LXI  H,X0412 LOAD HL WITH ADDR OF INTEGER OR SNG-PREC VALUE IN HOLDING AREA !  2189
0CBD  B7                   ORA  A  CLEAR CARRY BIT - IF VALUE IS DBL-PREC, DOES NOT JC AT 0CD7   7     2190
0CBE  3A A4 03   X0CBE     LDA  X03A4 LOAD ACC WITH TYPE-CODE OF VALUE IN HOLDING AREA           :$    2191
0CC1  3D                   DCR  A } DECREMENT ACC 3 TIMES - SETS FLAGS + TESTS TYPE-CODE         =     2192
0CC2  3D                   DCR  A }                                                              =     2193
0CC3  3D                   DCR  A }                                                              =     2194
0CC4  CA 1B 11   <----------JZ   X111B JMP IF VALUE IS STRING TYPE                               J     2195
0CC7  4E                   MOV  C,M } FETCH 2 BYTES OF VALUE FROM MEMORY - PUT THEM ON THE STACK N     2196
0CC8  23                   INX  H   }                                                            #     2197
0CC9  46                   MOV  B,M }                                                            F     2198
0CCA  C5                   PUSH B   }                                                            E     2199
0CCB  FA E6 0C   <----------JM   X0CE6 JMP IF VALUE IS INTEGER TYPE                                  2200
0CCE  23                   INX  H   } FETCH 2 MORE BYTES OF VALUE FROM MEMORY -                  #     2201
0CCF  4E                   MOV  C,M } PUT THEM ON THE STACK                                      N     2202
0CD0  23                   INX  H   }                                                            #     2203
0CD1  46                   MOV  B,M }                                                            F     2204
0CD2  C5                   PUSH B   }                                                            E     2205
0CD3  E2 E6 0C   <----------JPO  X0CE6 JMP IF VALUE IS SNG-PREC TYPE                                2206
0CD6  23                   INX  H   ADVANCE PTR TO VALUE                                         #     2207
0CD7  DA DD 0C             JC   X0CDD JMP IF DBL-PREC - ENTRY AT 0CBE (loc 0CBD)                 Z}    2208
0CDA  21 0F 04             LXI  H,X040E LOAD HL WITH ADDR OF LSB OF DBL-PREC HOLDING AREA        !     2209
0CDD  4E         X0CDD     MOV  C,M }                                                            N     2210
0CDE  23                   INX  H   } FETCH 4 MORE BYTES OF VALUE FROM MEMORY -                  #     2211
0CDF  46                   MOV  B,M }                                                            F     2212
0CE0  23                   INX  H   } PUT THEM ON STACK                                          #     2213
0CE1  C5                   PUSH B   }                                                            E     2214
0CE2  4E                   MOV  C,M }                                                            N     2215
0CE3  23                   INX  H   }                                                            #     2216
0CE4  46                   MOV  B,M }                                                            F     2217
0CE5  C5                   PUSH B   }                                                            E     2218
0CE6  DA 25 11   X0CE6     JC   X1125 JMP IF ENTERED AT 0CBE                                     Z%    2219
```

```
0CE9  C6 03          ADI  H'03' CORRECT TYPE CODE TO TRUE VALUE  (see ØCC1 - ØCC3)
0CEB  48             MOV  C,E PUT ARITHMETIC TOKEN - OFFSET IN C  (see .ØC91 or ØD18)
0CEC  47             MOV  B,A PUT VALUE TYPE CODE IN B
0CED  C5             PUSH B  PUT TOKEN-OFFSET + TYPE-CODE ON STACK
0CEE  01 2D 0D       LXI  B,X0D2D LOAD BC WITH THE ADDR OF THE ROUTINE THAT HANDLES */+-, AND CONVERTS THE ARGS TO THE APPROPRIATE PRECISIONS
0CF1  C5       X0CF1 PUSH B  PUT ADDR FOR IMPLIED JMP ON THE STACK
0CF2  2A CD 03       LHLD X03CD LOAD HL WITH ADDR OF LAST OPERATOR TOKEN FOUND - CONTINUE SCANNING FROM THE NEXT CHAR
0CF5  C3 55 0C       JMP  X0C55 CONTINUE PROCESSING THE STRING (ANOTHER VALUE, VAR., etc. THEN ANOTHER OPERATOR)
0CF8  CC 45 1C X0CF8 CALL X1C45 CONVERT VALUE IN HOLDING AREA TO SNG-PREC (THIS IS THE ARG TO BE RAISED TO SOME POWER)
0CFB  CD 36 1D       CALL X1D36 PUT THIS ARG ON THE STACK
0CFE  01 08 25       LXI  B,X2508 LOAD BC WITH THE ADDR OF EXPONENTIATION ROUTINE (USED AS AN IMPLIED JMP)
0D01  16 7F          MVI  D,H'7F' SET PRECEDENCE REG (PREVIOUS OPERATOR) TO 7F
0D03  C3 F1 0C       JMP  X0CF1
0D06  D5       X0D06 PUSH D SAVE CONTENTS OF DE ON THE STACK WHILE DOING THE NEXT ROUTINE
0D07  CD 11 1C       CALL X1C11 CONVERT THE VALUE IN HOLDING AREA TO INTEGER FORMAT - ARG1
0D0A  D1             POP  D RESTORE CONTENTS OF DE FROM STACK
0D0B  E5             PUSH H STORE INTEGER VALUE (FROM 1C11) ON THE STACK - ARG1
0D0C  01 02 0E       LXI  B,X0E02 LOAD BC WITH THE ADDR OF ROUTINE TO HANDLE 'MOD' & '\' ARITHMETIC
0D0F  C3 F1 0C       JMP  X0CF1
0D12  78       X0D12 MOV  A,B  } TEST PRECEDENCE OF PREVIOUS OPERATOR
0D13  FE 64          CPI  H'64' }
0D15  D0             RNC  - IF PREVIOUS OPERATOR(S) WERE ARITHMETIC - EVALUATE EXPRESSION TO LEFT OF CURRENT OPERATOR TO A SINGLE VALUE
0D16  C5             PUSH B SAVE PRECEDENCE OF PREVIOUS OPERATOR ON STACK
0D17  D5             PUSH D SAVE OFFSET OF ACCUMULATED OPERATORS ON STACK (see .ØC77)
0D18  11 04 64       LXI  D,X6404 LOAD E WITH TOKEN-OFFSET OF 4 (AFTER +-*/ IS COMPARISON ROUTINES - see ØIFB - Ø219)
0D1B  21 8C 0E       LXI  H,X0E8C } PUT ADDR ØE8C ON STACK FOR IMPLIED JMP.
0D1E  E5             PUSH H       }
0D1F  F7             RST  6 TEST VALUE TYPE OF ARG1
0D20  C2 BA 0C       JNZ  X0CBA IF ARITHMETIC VALUE FOR ARG1 - PUT ARG1 ON STACK, THEN IMPLIED JMP  (ADDR ØD7D (see ØCBA - ØCF1)
0D23  2A 12 04       LHLD X0412 } SAVE ADDR THAT POINTS TO 1ST CHAR OF 1ST ARG ON STACK (STRING TYPE ARG1)
0D26  E5             PUSH H     } CHAR COUNT BYTE OF IDENTITY GROUP
0D27  01 5F 0E       LXI  B,X0E5F LOAD BC WITH THE ADDR OF THE ROUTINE USED TO COMPARE 2 STRING VALUES FOR AN 'IF' STMT
0D2A  C3 F1 0C       JMP  X0CF1
0D2D  C1       X0D2D POP  B FETCH ARITHMETIC TOKEN OFFSET INTO REG C + VALUE TYPE CODE FOR ARG1 INTO REG B
0D2E  79             MOV  A,C } STORE ARITHMETIC TOKEN-OFFSET IN FLAG AREA (used by ØD79)
0D2F  32 A5 03       STA  X03A5 }
0D32  78             MOV  A,B  PUT VALUE TYPE CODE FOR ARG1 INTO ACC
0D33  FE 08          CPI  H'08' } IF ARG1 IS DBL-PREC, CONVERT ARG2 TO DBL-PREC & USE DBL-PREC FCN'S
0D35  CA 60 0D       JZ   X0D60 } (DBL-PREC */+-)
0D38  3A A4 03       LDA  X03A4 } IF ARG2 IS DBL-PREC, MOVE ARG2 TO TEMP STORAGE (Ø418-Ø41E)
0D3B  FE 08          CPI  H'08' } FETCH ARG1 FROM STACK INTO HOLDING AREA, CONVERT ARG1 TO DBL-PREC
0D3D  CA 88 0D       JZ   X0D88 } & USE DBL-PREC ARG'S
0D40  57             MOV  D,A PUT ARG2 TYPE-CODE IN D
0D41  78             MOV  A,B PUT ARG1 TYPE-CODE IN ACC
0D42  FE 04          CPI  H'04' } IF ARG1 IS SNG-PREC, CONVERT ARG2 TO SNG-PREC & USE SNG-PREC FCN'S
0D44  CA 9C 0D       JZ   X0D9C } (SNG-PREC */+-)
0D47  7A             MOV  A,D  PUT ARG2 TYPE-CODE IN ACC
0D48  FE 03          CPI  H'03' } IF ARG2 IS SNG-PREC, MOVE ARG1 TO HL, MOVE ARG2 TO STACK, CONVERT
0D4A  D2 A7 0D       JNC  X0DA7 } ARG1 TO SNG-PREC, PUT ARG1 IN BCDE, ARG2 IN HOLDING AREA, USE SNG-PREC FCN'S
0D4D  CA BA 1C       JZ   X1CBA IF ARG2 IS STRING TYPE (TYPE-CODE = 3) PRINT ERROR MESSAGE "TYPE MISMATCH"
0D50  21 0F 02       LXI  H,X020F LOAD HL WITH ADDR OF INTEGER FUNCTION TABLE
0D53  06 00          MVI  B,H'00' CLEAR B
0D55  09             DAD  B } COMPUTE ADDR OF ENTRY TO BE USED - PUT IN HL
0D56  09             DAD  B }
0D57  4E             MOV  C,M } FETCH ADDR OF FCN-ROUTINE FROM TABLE - PUT IN BC
0D58  23             INX  H
0D59  46             MOV  B,M }
0D5A  D1             POP  D  PUT ARG1 IN DE
0D5B  2A 12 04       LHLD X0412  PUT ARG2 IN HL
0D5E  C5             PUSH B  PUT ADDR OF FCN-ROUTINE ON STACK
                     RET  BRANCH TO FCN ROUTINE BY EXECUTING A RETURN
```

```
F     2220
K     2221
G     2222
E     2223
      2224
E     2225
M     2226
ER    2227
ME    2228
M6    2229
      2230
      2231
C     2232
U     2233
N     2234
Q     2235
      2236
      2237
C     2238
      2239
      2240
P     2241
E     2242
U     2243
      2244
!     2245
      2246
      2247
B     2248
      2249
      2250
      2251
C     2252
A     2253
      2254
2%    2255
      2256
      2257
J     2258
:3    2259
      2260
J     2261
H     2262
      2263
      2264
J     2265
      2266
      2267
      2268
      2269
!     2270
      2271
      2272
      2273
N     2274
#     2275
F     2276
Q     2277
*     2278
E     2279
```

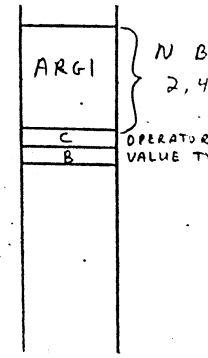
*/+-

Y↑X

AND OR MOD \

PRECEDENCE: < = 64 =

ARG1 FCN ARG2



ARG1 + ARG2 ARE BOTH INTEGER VALUES - USE INTEGER FCN'S

```
005F  C9                  RET                                                    I    2280
0060  CD 6F 1C  X0060    CALL  X1C6F CONVERT VALUE IN HOLDING AREA TO DBL-PREC (CDBL ROUTINE) - 2ND ARG  M    2281
0063  CD 91 18           CALL  X1891 MOVE DBL-PREC VALUE FROM (Ø4ØE-Ø415) TO (Ø418-Ø41F) - 2ND ARG     M    2282   DBL-PREC FCN DBL-PREC
0066  E1                  POP   H     } FETCH LO-4 BYTES                                                      2283
0067  22 10 04            SHLD  X0410 } OF DBL-PREC # FROM   } FETCH 8-BYTE DBL-PREC #                  "    2284
006A  E1                  POP   H     } STACK INTO HOLDING   } FROM STACK INTO HOLDING AREA                   2285   ARG1 FCN ARG2
006B  22 0E 04            SHLD  X040E } AREA                                                           "    2286          +
006E  C1           X006E  POP   B     } FETCH SNG-PREC or upper 4 BYTES of DBL PREC                  A    2287          -
006F  D1                  POP   D     } # FROM STACK INTO BCDE                                      Q    2288          *
0070  CD 46 18            CALL  X1846 STORE SNG-PREC VALUE IN BCDE INTO HOLDING AREA                 MF   2289          ÷
0073  CD 6F 1C  X0073    CALL  X1C6F CONVERT VALUE IN HOLDING AREA TO DBL-PREC (CDBL ROUTINE)           M    2290
0076  21 F8 01           LXI   H,X01F8 LOAD HL WITH ADDR OF DBL-PREC FUNCTION TABLE                     !    2291   FCN'S WANT ARG1 IN
0079  3A A5 03  X0079    LDA   X03A5 LOAD ACC WITH OFFSET OF ARITHMETIC FCN TO PERFORM  } COMPUTE       1%   2292
007C  07                  RLC   MULTIPLY OFFSET BY 2 (ADDR TABLE - 2 BYTES PER ENTRY)  } ADDR OF        2293   Ø4ØE-Ø415 + ARG2 IN
007D  C5                  PUSH  B SAVE BC ON STACK SINCE BC USED IN THE NEXT COMPUTATION } FCN ROUTINE  E    2294   Ø418-Ø41F
007E  4F                  MOV   C,A PUT TABLE-OFFSET IN C                                              O    2295
007F  06 00               MVI   B,H'00' CLEAR B                                                             2296
0081  09                  DAD   B COMPUTE ADDR OF ENTRY TO BE USED - PUT IN HL                              2297
0082  C1                  POP   B RESTORE BC FROM STACK                                                A    2298
0083  7E                  MOV   A,M } FETCH ADDR OF FCN-ROUTINE FROM TABLE                                  2299
0084  23                  INX   H   }                                                                  #    2300
0085  66                  MOV   H,M }                                                                       2301
0086  6F                  MOV   L,A }                                                                       2302
0087  E9                  PCHL      BRANCH TO FCN-ROUTINE                                                   2303
0088  C5           X0088  PUSH  B  SAVE TYPE-CODE FOR 1ST ARG ON STACK                                  E    2304   CONVERT ARG1 TO DBL-PREC
0089  CD 91 18            CALL  X1891 MOVE DBL-PREC VALUE FROM (Ø4ØE-Ø415) TO (Ø418-Ø41F) - 2ND ARG    M    2305
008C  F1                  POP   PSW FETCH TYPE CODE FOR 1ST ARG FROM STACK INTO ACC                        2306
008D  32 A4 03            STA   X03A4 PUT TYPE-CODE IN FLAG AREA                                       2$   2307
0090  FE 04               CPI   H'04' } IF 2ND ARG IS SNG-PREC, JMP TO Ø0BE, FETCH 4-BYTES FROM STACK      2308   } FETCH 2 or 4 BYTES FOR ARG1
0092  CA 6E 00            JZ    X006E }                                                                J    2309   } FROM STACK - THEN CONVERT
0095  E1                  POP   H     } OTHERWISE, TYPE-CODE IS INTEGER TYPE, FETCH 2-BYTES FROM STACK      2310   } ARG1 TO DBL-PREC - USE DBL-PREC
0096  22 12 04            SHLD  X0412 }                                                                "    2311   } FCN'S */+-
0099  C3 73 00            JMP   X0073                                                                  C    2312   }
009C  CD 45 1C  X009C    CALL  X1C45 CONVERT VALUE IN HOLDING AREA TO SNG-PREC (CSGN ROUTINE) - 2ND  MEARG 2313   SNG-PREC
009F  C1                  POP   B } FETCH SNG-PREC # FROM STACK - 1ST ARG                               A    2314
00A0  D1                  POP   D }                                                                     Q    2315   ARG1 FCN ARG2
00A1  21 05 02     X00A1  LXI   H,X0205 LOAD HL WITH ADDR OF SNG-PREC FUNCTION TABLE                    !    2316   FCN'S WANT ARG1 IN BCDE
00A4  C3 79 00            JMP   X0079 COMPUTE ADDR FROM TABLE & BRANCH TO FCN TABLE                     C    2317          & ARG2 IN Ø412-Ø415
00A7  E1           X00A7  POP   H  FETCH 2-BYTE INTEGER # FROM STACK - 1ST ARG                               2318   CONVERT ARG1 TO SNG-PREC
00A8  CD 36 18            CALL  X1836 PUT SNG-PREC # (IN HOLDING AREA) ONTO STACK - 2ND ARG             M6   2319
00AB  CD 63 1C            CALL  X1C63 CONVERT # IN HL TO SNG-PREC - 1ST ARG                             M    2320
00AE  CD 51 18            CALL  X1851 PUT CONVERTED # IN BCDE (MOVE FROM HOLDING AREA) - 1ST ARG        MQ   2321   ARG1 FCN ARG2
00B1  E1                  POP   H     } FETCH SNG-PREC # FROM STACK INTO                                    2322
00B2  22 14 04            SHLD  X0414 } HOLDING AREA - 2ND ARG                                         "    2323   CONVERT ARGS TO SNG-PREC
00B5  E1                  POP   H     }                                                                     2324   USE SNG-PREC FCN'S
00B6  22 12 04            SHLD  X0412 }                                                                "    2325
00B9  C3 A1 00            JMP   X00A1 COMPUTE ADDR FROM TABLE & BRANCH TO FCN TABLE                     C!   2326
00BC  E5                  PUSH  H SAVE # IN HL ON STACK (2ND #)                                             2327   INTEGER DIVIDE
00BD  EB                  XCHG  MOVE # IN DE TO HL (1ST #)                                                  2328   (# IN DE) ÷ (# IN HL)
00BE  CD 63 1C            CALL  X1C63 CONVERT # IN HL FROM INTEGER FORMAT TO SNG-PREC VALUE IN HOLDING-MAREA 2329   (1ST #)  (2ND #)
00C1  E1                  POP   H FETCH 2ND # FROM STACK INTO HL                                            2330
00C2  CD 36 18            CALL  X1836 PUT # IN HOLDING AREA ONTO STACK (1ST #)                          M6   2331
00C5  CD 63 1C            CALL  X1C63 CONVERT # IN HL TO SNG-PREC VALUE IN HOLDING AREA (2ND #)         M    2332
00C8  C3 2C 1A            JMP   X1A2C JMP TO SNG-PREC DIVIDE ROUTINE                                    C,   2333
00CB  D7           X00CB  RST   2     SCAN TO NXT CHAR  RC = DIGIT                                       K    2334   PREFIX-CHAR EVALUATOR
00CC  DA 43 20            JC    X2043 JMP IF CHAR IS DIGIT - CONVERT STRING TO NUMERIC VALUE            ZC   2335   (EVALUATE 1 QUANTITY IN STRING TO
00CF  CD E1 08            CALL  X08E1 TEST CHAR  RNC A-Z                                                M    233E   A VALUE IN HOLDING AREA)
00D2  D2 0E 0E            JNC   X0E0E IF CHAR IS A-Z, STRING IS VARIABLE NAME, FETCH VARIABLE VALUE FROM R SYMBOL TABLE INTO HOLDING AREA  2337
00D5  FE AE               CPI   H'AE' } JMP IF CHAR IS '+' ('+'TOKEN IS AE) - LEADING '+' IS            •    2338   IGNORED
00D7  CA CB 00            JZ    X00CB }                                                                 JK   2339
```

```
0D0A  FE 2E       CPI  A'.'   } JMP. IF CHAR '.', NXT GROUP OF CHAR'S SHOULD BE DIGITS
0D0C  CA 43 20    JZ   X2043  }
0D0F  FE AF       CPI  H'AF'  } JMP IF CHAR '-', NEGATE NXT PART OF STRING TO BE EVALUATED
0D11  CA 00 0E    JZ   X0E00  }
0D14  FE 22       CPI  A'"'   } JMP IF CHAR '"', EVALUATE STRING TO STRING-TYPE VALUE
0D16  CA FD 11    JZ   X11FD  }
0D19  FE AC       CPI  H'AC'  } JMP IF CHAR IS 'NOT' TOKEN, PERFORM 'NOT' FCN ON NXT PART OF
0D1B  CA 99 0E    JZ   X0E99  } STRING TO BE EVALUATED
0D1F  FE A8       CPI  H'A8'  } JMP IF CHAR IS 'FN' TOKEN, EVALUATE USER FCN
0D20  CA E8 10    JZ   X10E8  }
0D23  D6 BA       SUI  H'BA'  } REDUCE TOKEN TO AN OFFSET
0D25  D2 1C 0E    JNC  X0E1C  } JMP IF CHAR IS 'SGN' TOKEN or LARGER - TEST FOR SINGLE-ARG FUNCTION TOKEN
0D28  CF          X0DF8 RST  1     } SYNTAX CHECK. CHAR MUST BE '('
0D29  28          DB '('  ~~DATA A'('~~ }
0D2A  CC 52 0C    CALL X0C52 RECURSIVE CALL - EVALUATE STRING WITHIN PARENTHESES
0D2D  CF          RST  1     } SYNTAX CHECK. CHAR MUST BE ')'
0D2E  29          DB ')'  DAD  H
0D2F  C9          RET  DONE

0E00  16 70       X0E00 MVI  D,H'70' SET PRECEDENCE OF PREVIOUS OPERATOR TO 70 (LARGER THAN * / & LESS THAN ↑)
0E02  CD 55 0C    CALL X0C55 EVALUATE NEXT PORTION OF STRING (UNTIL OPERATOR LESS THAN 70 IS FOUND - THEN NEGATE THE FIRST VALUE)
0E05  2A 05 03    LHLD X0305 LOAD HL WITH ADDR OF NEXT CHAR IN STRING TO BE PROCESSED
0E08  E5          PUSH H     SAVE PTR TO STRING ON STACK
0E09  CD 0C 18    CALL X180C NEGATE THE VALUE IN THE HOLDING AREA
0E0C  E1          X0E0C POP  H     FETCH PTR TO STRING FROM STACK
0E0D  C9          RET        DONE

0E0E  CD E0 0E    X0E0E CALL X0EE0 FIND VARIABLE IN SYMBOL TABLE - STMT PTR SCANS PAST NAME OF VARIABLE
0E11  E5          X0E11 PUSH H SAVE PTR TO STMT ON STACK
0E12  EB          XCHG  PUT ADDR OF VARIABLE VALUE REGION IN HL (WAS IN DE - see 0FA3)
0E13  22 12 04    SHLD X0412 PUT ADDR OF VARIABLE REGION IN LOC'S 0412-0413 - USEFUL IF STRING-TYPE
0E16  F7          RST  6     TEST TYPE OF VARIABLE
0E17  C4 BB 18    CNZ  X18BB IF ARITHMETIC TYPE OF VARIABLE, CALL 18BB, MOVE VALUE FROM SYMBL
0E1A  E1          POP  H FETCH PTR TO STMT FROM STACK
0E1B  C9          RET  DONE

0E1C  06 00       X0E1C MVI  B,H'00' CLEAR REG B
0E1E  07          RLC        MULTIPLY TOKEN OFFSET BY 2
0E1F  4F          MOV  C,A   PUT ADJUSTED OFFSET IN C
0E20  C5          PUSH B     SAVE ADJUSTED SINGLE-ARG FCN TOKEN ON STACK
0E21  D7          RST  2     SCAN TO NEXT NON-SPACE CHAR IN STRING TO BE EVALUATED
0E22  79          MOV  A,C   } TEST ADJUSTED TOKEN OFFSET
0E23  FE 2F       CPI  A'/'  }
0E25  DA 41 0E    JC   X0E41 JMP IF TOKEN IS SGN TO CHR$
0E28  CF          RST  1     } SYNTAX CHECK FOR '('
0E29  28          DATA A'('  }
0E2A  CD 52 0C    CALL X0C52 EVALUATE 1ST ARG - SHOULD BE STRING-TYPE VALUE
0E2D  CF          RST  1     } SYNTAX CHECK FOR ','
0E2E  2C          INR  L     }
0E2F  CD BB 1C    CALL X1CBB PERFORM STRING VALUE CHECK ON VALUE IN HOLDING AREA
0E32  EB          XCHG       SAVE PTR TO STRING TO BE EVALUATED IN DE
0E33  2A 12 04    LHLD X0412 PUT PTR TO CHAR COUNT BYTE OF 1ST ARG STRING
0E36  E3          XTHL       } PUT PTR TO CHAR COUNT BYTE ON STACK ABOVE FCN-TOKEN (see 0E20)
0E37  E5          PUSH H     }
0E38  EB          XCHG       PUT PTR TO STRING TO BE EVALUATED IN HL
0E39  CD 81 14    CALL X1481 EVALUATE 2ND ARG. TO 1 BYTE INTEGER VALUE, VALUE IN E
0E3C  E3          XCHG } PUT VALUE FOR 2ND ARG ON STACK, PTR TO STRING TO BE EVALUATED IN DE
0E3D  EB          XTHL } PUT FCN-TOKEN IN HL
0E3E  C3 56 0E    JMP  X0F56 COMPUTE ADDR OF FCN ROUTINE, BRANCH TO THAT ADDRESS TO EXECUTE THE FCN

0E41  CD F8 0D    X0E41 CALL X0DF8 PERFORM SYNTAX CHECK FOR '(', EVALUATE ARG, PERFORM SYNTAX CHECK FOR ')'
0E44  E3          XTHL PUT PTR TO STRING ON STACK, PUT ADJUSTED TOKEN OFFSET IN L (see 0E1C - 0E20)
0E45  7D          MOV  A,L   } TEST ADJUSTED TOKEN OFFSET
0E46  FE 0E       CPI  H'0E' }
```

2340 CONVERT STRING TO NUMERIC VALUE.
2341
2342
2343
2344
2345
2346 STRING TO BE EVALUATED
2347
2348
2349
2350 ARG FUNCTION TOKEN
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361 ie $-A+B \Rightarrow -(A)+B$
2362 $-A \uparrow B \Rightarrow -(A \uparrow B)$
2363
2364
2365
2366 FETCH VARIABLE VALUE FROM SYMBOL
2367 TABLE FOR USE IN A COMPUTATION
2368 VARIABLE
2369
2370 TABLE TO HOLDING AREA
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381 LEFT$
2382 RIGHT$
2383 MID$
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393 DE
2394
2395
2396
2397
2398
2399

```
0E48  DA 52 0E          JC    X0E52 JMP IF FCN IS SGN TO POS                    ZR   2400
0E4B  FE 10             CPI   H'10' TEST ADJUSTED TOKEN OFFSET                     2401
0E4D  E5                PUSH  H SAVE HL ON STACK SINCE HL MIGHT BE USED IF CC 1C45 at ØE4E  2402
0E4E  DC 45 1C          CC    X1C45 CONVERT ARG IN HOLDING AREA TO SNG-PREC IF FCN IS SQR TO ATN\E  2403
0E51  E1                POP   H RESTORE HL FROM STACK (ADJUSTED TOKEN OFFSET)     2404
0E52  11 0C 0E  X0E52   LXI   D,X0E0C } PUT ADDR ØE0C ON STACK FOR IMPLIED JMP    2405
0E55  D5                PUSH  D       } - AFTER EXECUTING FCN ROUTINE, PUT PTR TO STRING IN HL, THEN U  2406 RETURN TO ØC5E
0E56  01 3B 00  X0E56   LXI   B,X003B LOAD BC WITH ADDR OF START OF SINGLE-ARG FCN TABLE  ;  2407
0E59  09        X0E59   DAD   B  COMPUTE ADDR OF FCN ROUTINE ADDR IN TABLE - PUT IN HL  2408
0E5A  4E                MOV   C,M } FETCH ADDR OF FCN ROUTINE INTO HL             N   2409
0E5B  23                INX   H                                                   #   2410
0E5C  66                MOV   H,M                                                     2411
0E5D  69                MOV   L,C }                                                   2412
0E5E  E9                PCHL      BRANCH TO ROUTINE TO PERFORM FCN                    2413
0E5F  CD 77 13  X0E5F   CALL  X1377 REMOVE IDENTITY GROUP OF 2ND ARG FROM FORMULA TABLE, REMOVE STRING FROM M STRING  2414 AREA
0E62  7E                MOV   A,M PUT CHAR COUNT BYTE OF 2ND ARG IN ACC               2415 COMPARISON OF 2 STRINGS FOR
0E63  23                INX   H } PUT ADDR OF 1ST CHAR OF 2ND ARG STRING IN BC  #   2416
0E64  4E                MOV   C,M                                             N   2417 AN IF STMT
0E65  23                INX   H                                               #   2418
0E66  46                MOV   B,M }                                           F   2419
0E67  D1                POP   D   PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN DE (see ØD23 - ØD26)  2420
0E68  C5                PUSH  B   PUT PTR TO 1ST CHAR OF 2ND ARG STRING ON STACK  E  2421
0E69  F5                PUSH  PSW PUT CHAR COUNT OF 2ND ARG ON STACK                  2422
0E6A  CD 7E 13          CALL  X137E REMOVE IDENTITY GROUP OF 1ST ARG FROM FORMULA TABLE, REMOVE STRING FROM M  2423 STRING AREA
0E6D  D1                POP   D   PUT CHAR COUNT BYTE OF 2ND ARG IN D             Q   2424
0E6E  5E                MOV   E,M PUT CHAR COUNT BYTE OF 1ST ARG IN E (HL POINTS TO CHAR COUNT BYTE OF E TO CALL 137E AT ØE6A)  2425
0E6F  23                INX   H } PUT PTR TO 1ST CHAR OF 1ST ARG IN BC        #   2426
0E70  4E                MOV   C,M                                             N   2427
0E71  23                INX   H                                               #   2428
0E72  46                MOV   B,M                                             F   2429
0E73  E1                POP   H   PUT PTR TO 1ST CHAR OF 2ND ARG IN HL                2430
0E74  78        X0E74   MOV   A,E } TEST IF BOTH CHAR CTR'S ARE Ø                    2431
0E75  B2                ORA   D }                                             Z   2432 RESULT IN ACC
0E76  C8                RZ        EXIT IF BOTH CHAR CTR'S ARE Ø               H   2433
0E77  7A                MOV   A,D PUT CHAR CTR OF 2ND ARG IN ACC                      2434 ARG1 = ARG2   ACC Ø   CARRY Ø
0E78  D6 01             SUI   H'01' } RETURN IF 2ND ARG CHAR COUNT IS ZERO     V   2435
0E7A  D8                RC        }                                           X   2436 ARG1 > ARG2   -1      1
0E7B  AF                XRA   A   CLEAR ACC                                   /   2437
0E7C  BB                CMP   E   }                                           ;   2438 ARG1 < ARG2   1       Ø
0E7D  3C                INR   A   }                                           <   2439
0E7E  D0                RNC       } RETURN IF 1ST ARG CHAR COUNT IS ZERO       P   2440 NOTE: THESE MATCH CONDITIONS
0E7F  15                DCR   D } DECREMENT CHAR CTR'S FOR BOTH ARGS                  2441 DOCUMENTED IN ØE3C
0E80  1D                DCR   E }                                                     2442
0E81  0A                LDAX  B FETCH A CHAR FROM 1ST ARG STRING INTO ACC             2443
0E82  BE                CMP   M COMPARE AGAINST A CHAR IN 2ND ARG STRING      >   2444
0E83  23                INX   H ADVANCE CHAR POSITION PTR TO 2ND ARG STRING   #   2445
0E84  03                INX   B ADVANCE CHAR POSITION PTR TO 1ST ARG STRING           2446
0E85  CA 74 0E          JZ    X0E74 IF THE CHAR'S MATCH, LOOP TO TEST THE NEXT PAIR OF CHAR'S IN THE STRINGS  2447 FOR A MATCH
0E88  3F                CMC       COMPLEMENT CARRY BIT, CARRY 1ST > 2ND, NO CARRY 1ST < 2ND   TO ?MATCH  2448 RESULTS IN TABLE
0E89  C3 F1 1A          JMP   X1AF1 PUT RESULT IN ACC, IF 1ST > 2ND A.C. = -1 CARRY SET, IF 1ST < 2ND  C  ACC=1  2449 CARRY NOT SET
0E8C  3C        X0E8C   INR   A } ADJUST RESULT OF MAGNITUDE TEST             <   2450
0E8D  8F                ADC   A }                                                     2451 AFTER DOING ARG1 COMPARE ARG2 (MAG
0E8E  C1                POP   B FETCH RELATIONAL OPERATOR MASK FROM STACK (see ØC72-ØC77)  A  2452 COMPARISON) TEST IF TRUE OR FALSE
0E8F  A0                ANA   B AND MASK WITH RESULT OF TEST                         2453 QUANTITY IN STRING BEING EVALUATED
0E90  CE FF             ADI   H'FF' ADJUST RESULT                             F   2454 RESULT OF COMPARISON TEST
0E92  9F                SBB   A } TRUE = -1 FALSE = Ø CARRY SET IF TRUE              2455 ARG1 ? ARG2
0E93  CD 1E 1B          CALL  X1B1E CONVERT TRUE OR FALSE VALUE TO A 2 BYTE INTEGER - STORE IN HOLDING AREA  M  2456  > ACC = -1
0E96  C3 61 0C          JMP   X0C61 JMP TO ØC61 TO PROCESS PRESENT OPERATOR   C   2457  = ACC = Ø
0E99  16 5A     X0E99   MVI   D,A'Z' SET PRECEDENCE OF PREVIOUS OPERATOR TO 5A (LARGER THAN 'AND', LESS THAN + -)  Z  2458  < ACC = 1
0E9B  CD 55 0C          CALL  X0C55 EVALUATE NEXT PORTION OF STRING (UNTIL OPERATOR LESS THAN 5A IS FOUND - MU  THEN  2459 NOT THE FIRST VALUE)
```

ARG1 = ARG2 → RZ (ACC = Ø)
ARG1 > ARG2 → SUI H'01' / RC (ACC = -1)
ARG1 < ARG2 → INR A / RNC (ACC = 1)
STRINGS STILL MATCH → JZ X0E74

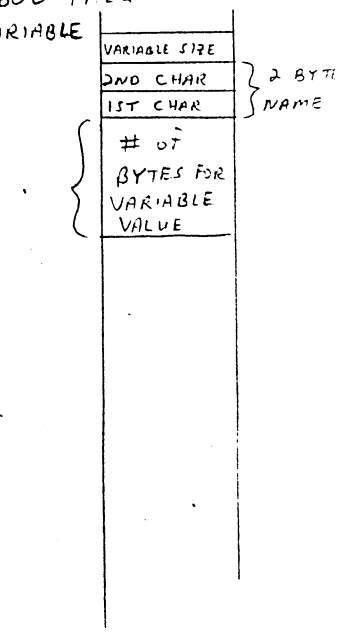
> = < : -1 Ø 1 → Ø 1NC 2NC → 1 2 → 4 → ACC=Ø IF FALSE, ACC=1 IF TRUE

NOT

```
0F9E  CD 11 1C          CALL X1C11 CONVERT VALUE IN HOLDING AREA TO A 16-BIT INTEGER VALUE - IN HL + 2460 HOLDING AREA
0EA1  7D                MOV  A,L  } COMPLEMENT VALUE                                              2461
0EA2  2F                CMA       }                                                               2462
0EA3  6F                MOV  L,A  }                                                               2463
0EA4  7C                MOV  A,H  }                                                               2464
0EA5  2F                CMA       }                                                               2465
0EA6  67                MOV  H,A  }                                                               2466
0EA7  22 12 04          SHLD X0412 STORE COMPLEMENTED INTO HOLDING AREA                           2467
0EAA  C1                POP  B REMOVE RETURN ADDR FROM STACK (ØC5E).                              2468
0EAB  C3 61 0C          JMP  X0C61 JMP TO ØC61 TO PROCESS PRESENT OPERATOR                        2469
-------------------------------------------------------------------------------------------------------
0EAE  3D          X0EAE DCR  A } DECREMENT ACC 3 TIMES                                            2470 END OF RST 6
0EAF  3D                DCR  A } SETS ZERO FLAG, DOES NOT AFFECT CARRY FLAG                       2471 RZ - CHAR STRING VARIABLE
0EB0  3D                DCR  A }                                                                  2472 RNC - DOUBLE PRECISION
0EB1  C9                RET                                                                       2473 RP - SNG PRECISION   RM - INTEGER
-------------------------------------------------------------------------------------------------------
0EB2  C5          X0EB2 PUSH B    SAVE PREVIOUS OPERATOR PRECEDENCE ON STACK                       2474
0EB3  CD 11 1C          CALL X1C11 CONVERT ARG2 TO INTEGER FORMAT - IN HL & HOLDING AREA           2475
0EB6  F1                POP  PSW FETCH PREVIOUS OPERATOR PRECEDENCE INTO ACC FROM STACK           2476
0EB7  D1                POP  D   FETCH ARG1 FROM STACK (see ØDØB)                                 2477
0EB8  FE 7A             CPI  H'7A' } IF PREVIOUS OPERATOR IS 7A, PERFORM ARG1 MOD ARG2            2478  ie A\B + C
0EBA  CA 31 1E          JZ   X1E31 }                                                              2479        ↑   ↑
0EBD  FE 7B             CPI  H'7B' } IF PREVIOUS OPERATOR IS 7B, PERFORM ARG1 \ ARG2               2480   previous  present
0EBF  CA CC 1D          JZ   X1DCC }                                                              2481      op      operator
0EC2  FE 46             CPI  A'F' TEST PREVIOUS OPERATOR PRECEDENCE BYTE                           2482
0EC4  7B                MOV  A,E PUT LO-ORDER BYTE OF ARG1 INTO ACC (USED BY 'AND' or 'OR' ROUTINE - 2483 SAVES 1 INSTRUCTION)
0EC5  CA CF 0E          JZ   X0ECF JMP IF DOING 'OR' FUNCTION                                     2484
0EC8  A5                ANA  L  } AND LO-ORDER BYTES OF ARG1 + ARG2                               2485 ↓ AND
0EC9  6F                MOV  L,A }                                                                2486
0ECA  7C                MOV  A,H } AND HI-ORDER BYTES OF ARG1 + ARG2                              2487
0ECB  A2                ANA  D  }                                                                 2488
0ECC  C3 D6 10          JMP  X10D6 STORE RESULT IN HOLDING AREA (INTEGER FORMAT VALUE)            2489
0ECF  B5          X0ECF ORA  L  } OR LO-ORDER BYTES OF ARG1 + ARG2                                2490 ↓ OR
0ED0  6F                MOV  L,A }                                                                2491
0ED1  7C                MOV  A,H } OR HI-ORDER BYTES OF ARG1 + ARG2                               2492
0ED2  B2                ORA  D  }                                                                 2493
0ED3  C3 D6 10          JMP  X10D6 STORE RESULT IN HOLDING AREA (INTEGER FORMAT VALUE)            2494
-------------------------------------------------------------------------------------------------------
0ED6  2B          X0ED6 DCX  H                                                                    2495   PART OF DIM
0ED7  D7                RST  2 } TEST NXT CHAR, IF NULL OR ':' RETURN                             2496   STRING SCAN LOOP
0ED8  C8                RZ     }                                                                  2497   FOR NEXT ARRAY TO DIM
0ED9  CF                RST  1 } OTHERWISE NXT CHAR MUST BE A COMMA                               2498
0EDA  2C                INR  L DB ',' } SYNTAX ERROR IF NOT A COMMA                               2499
-------------------------------------------------------------------------------------------------------
0EDC  01 D6 0E          LXI  B,X0ED6 } PUT ADDR ON STACK (PROVIDES LOOPING BY USE OF              2500   DIM
0EDF  C5                PUSH B        }   RETURN - IMPLIED JMP)                                   2501
0EE0  F6 AF       STRING EVALUATOR 'ERASE' & 'LET' ENTRY POINT
                        ORI  H'AF' DUMMY XRA A } REMOTE ENTRY CLEAR ACC + LOC Ø3A3                 2502
0EE2  32 A3 03          STA  X03A3                                                                2503
0EE5  46                MOV  B,M  PUT FIRST CHAR OF NAME IN B REG                                 2504
0EE6  CD E1 08    X0EE6 CALL X08E1 TEST CHAR - PUTS CHAR IN ACC FROM MEMORY FOR THE TEST          2505
0EE9  DA A9 04          JC   X04A9 PRINT MESSAGE "SYNTAX ERROR" IF NOT A-Z (FIRST CHAR            2506 OF MATRIX NAME)
0EEC  AF                XRA  A  } CLEAR ACC                                                       2507
0EED  4F                MOV  C,A } & C REG (2ND CHAR IS A NULL IF NOT A-Z OR A DIGIT)             2508
0EEE  D7                RST  2 POINT TO NXT NON-SPACE CHAR (2ND CHAR OF NAME)                     2509
0EEF  DA F7 0E    DIGIT JC   X0EF7 JMP IF 2ND CHAR IS A DIGIT.  {IF ANY}                          2510
0EF2  CD E1 08          CALL X08E1 TEST CHAR (POINTED TO BY HL)                                   2511
0EF5  DA 02 0F  NOT A-Z JC   X0F02 JMP IF NOT A-Z                                                 2512
0EF8  4F                MOV  C,A PUT 2ND CHAR OF NAME IN C                                        2513
0EF9  D7          X0EF8 RST  2 ADVANCE PTR TO NXT NON-SPACE CHAR                                  2514
0EFA  DA F8 0E  ↑ DIGIT JC   X0EF8 JMP IF NXT CHAR IS DIGIT } LOOP ADVANCES HL PAST               2515
0EFD  CD E1 08          CALL X08E1 TEST CHAR                 } DIGITS + A-Z CHAR                  2516
0F00  D2 F8 0E  A-Z     JNC  X0EF8 JMP IF A-Z                } (IGNORE ANY OVER 2 CHAR IN         2517 ARRAY NAME)
0F03  11 29 0F    X0F02 LXI  D,X0F29 } PUT ØF29 ON STACK (TO BE USED WITH RZ OR RET)              2518
0F06  D5                PUSH D        }   AS A JMP                                                2519
```

```
                        NXT TEST FOR ARRAY DEFINITION
0F06  16 02             MVI  D,H'02'  } SET D=2  IF
0F08  FE 25             CPI  A'%'     } INTEGER TYPE
0F0A  C8                RZ   (JMPS TO ØF29)
0F0B  14                INR  D        } SET D=3 IF STRING TYPE
0F0C  FE 24             CPI  A'$'     }
0F0E  C8                RZ   (JMPS TO ØF29)
0F0F  14                INR  D        } SET D=4 IF SINGLE PRECISION TYPE
0F10  FE 21             CPI  A'!'     }
0F12  C8                RZ   (JMPS TO ØF29)
0F13  16 08             MVI  D,H'08'  } SET D=8 IF DOUBLE PRECISION TYPE
0F15  FE 23             CPI  A'#'     }
0F17  C8                RZ   (JMPS TO ØF29)
0F18  78                MOV  A,B  PUT FIRST CHAR OF NAME IN ACC   } FETCH "SIZE" OF
0F19  D6 41             SUI  A'A' SUBTRACT OFFSET                 } VARIABLE FROM
0F1B  E6 7F             ANI  H'7F' DELETE D7 BIT                  } DEFAULT TABLE
0F1D  5F                MOV  E,A  SAVE THIS VALUE IN REG E        }
0F1E  16 00             MVI  D,H'00' CLEAR REG D                  }
0F20  E5                PUSH H    PUT PTR TO STMT ON STACK
0F21  21 EA 03          LXI  H,X03EA LOAD ADDR OF TABLE
0F24  19                DAD  D    ADD  OFFSET
0F25  56                MOV  D,M  FETCH BYTE FROM TABLE (DEFAULT SIZE OF VARIABLE)
0F26  E1                POP  H    RESTORE PTR FROM STACK
0F27  2B                DCX  H    BACK UP PTR BY 1 (POINT TO CHAR BEFORE '(')
0F28  C9                RET  (JMP TO ØF29).

                        SCAN SYMBOL TABLE
                        TO FIND VARIABLE
0F29  7A        X0F29   MOV  A,D  } STORE VARIABLE SIZE IN LOC Ø3A4
0F2A  32 A4 03          STA  X03A4}
0F2D  D7                RST  2  ADVANCE PTR TO POINT TO '(' IF ARRAY NAME OR CHAR AFTER NAME
0F2E  3A 01 03          LDA  X0301 "NEW" SETS Ø3D1 TO ØØ - TEMP STORAGE - ERASE SETS Ø3D1 TO 1
                                                                  FOR  SETS Ø3D1 TO 64
0F31  3D                DCR  A
0F32  CA E1 0F          JZ   X0FE1 JMP IF ERASE COMMAND IS BEING PROCESSED   (ERASE)
0F35  86                ADD  M    ADD CHAR '(' TO ACC
0F36  D6 27             SUI  A'(' SUB '(' FROM ACC (1 LESS THAN '(')
0F38  CA BB 0F          JZ   X0FBB JMP IF CHAR IS '('   (SCAN ARRAY TABLE)
0F3B  AF                XRA  A    } CLEAR LOC Ø3D1
0F3C  32 01 03          STA  X0301}
0F3F  E5                PUSH H  SAVE PTR TO STMT ON STACK
0F40  D5                PUSH D  SAVE VARIABLE "SIZE" ON STACK (REG D)
0F41  2A E1 03          LHLD X03E1 START SCAN WITH PTR SET WITH ADDR OF IST BYTE OF SYMBOL TABLE
0F44  EB        X0F44   XCHG  SAVE PTR TO TABLE IN DE
0F45  2A E3 03          LHLD X03E3 FETCH PTR FROM Ø3E3 + Ø3E4 (START ADDR OF TABLE FOLLOWING ARRAY TABLE)
0F48  E7                RST  4  RZ  HL = DE
0F49  E1                POP  H  PUT VARIABLE SIZE IN H
0F4A  CA 69 0F          JZ   X0F69  END OF LOOP WHEN ENTIRE SYMBOL TABLE HAS BEEN SCANNED
                                                  (VARIABLE NOT FOUND)
0F4D  1A                LDAX D  FETCH BYTE FROM TABLE
0F4E  6F                MOV  L,A  SAVE BYTE IN L (THIS IS THE SIZE OF VARIABLE BEING SCANNED IN THE TABLE)
0F4F  BC                CMP  H  COMPARE WITH VARIABLE SIZE OF VARIABLE IN STMT
0F50  13                INX  D  ADVANCE PTR TO TABLE
0F51  C2 60 0F          JNZ  X0F60 JMP IF NO MATCH
0F54  1A                LDAX D  FETCH NEXT BYTE FROM TABLE
0F55  B9                CMP  C  COMPARE WITH SECOND CHAR OF VARIABLE NAME
0F56  C2 60 0F          JNZ  X0F60 JMP IF NO MATCH
0F59  13                INX  D  ADVANCE PTR TO TABLE
0F5A  1A                LDAX D  FETCH BYTE FROM TABLE
0F5B  B8                CMP  B  COMPARE WITH FIRST CHAR OF VARIABLE NAME
0F5C  CA A3 0F          JZ   X0FA3 JMP IF MATCH (BOTH CHARS OF NAME MATCH + SIZE DEFN MATCHES)
                                                  (VARIABLE FOUND)
0F5F  3E 13     INX D   MVI  A,H'13' Dummy
0F61  13                INX  D  ADVANCE PTR TO TABLE (AT THIS POINT, PTR POINTS TO BYTE AFTER
0F62  E5                PUSH H  PUT VARIABLE SIZE ON STACK (H = SIZE OF VARIABLE IN DIM STMT,
0F63  26 00             MVI  H,H'00' CLEAR H  } COMPUTE ADDR OF NXT VARIABLE IN TABLE
0F65  19                DAD  D  ADD PTR TO TABLE }
```

	2520
%	2521
H	2522
	2523
$	2524
H	2525
	2526
!	2527
H	2528
	2529
#	2530
H	2531
	2532
VA	2533
	2534
-	2535
	2536
	2537
!	2538
	2539
V	2540
	2541
+	2542
I	2543
	2544
2$	2545
H	2546
:Q	2547
=	2548
J	2549
	2550
V'	2551
J:	2552
/	2553
2Q	2554
	2555
U	2556
	2557
	2558
	2559
	2560
	2561
J	2562
	2563
	2564
<	2565
	2566
B	2567
	2568
9	2569
8	2570
	2571
	2572
8	2573
)#	2574
>	2575
	2576
	2577
&	2578
	2579

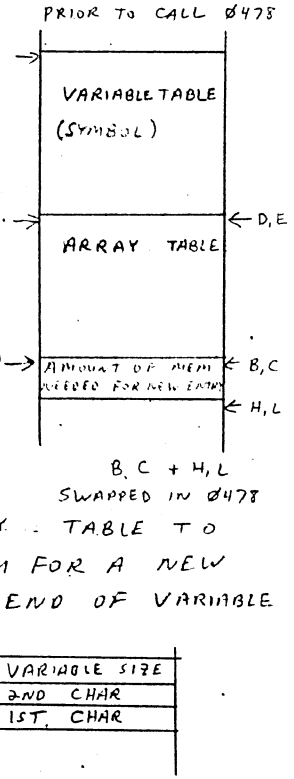


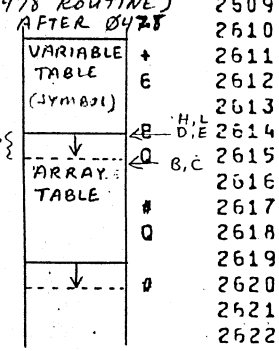
```
0F66  C3 44 0F          JMP   X0F44   ENTER HERE IF SYMBOL_NAME (+ SIZE DEFN) NOT FOUND IN CD
0F69  7C          X0F69 MOV   A,H     PUT VARIABLE SIZE IN ACC (VARIABLE IN DIM STMT)
0F6A  F1 VARIABLE NOT FOUND IN TABLE  POP   H     LEAVE PTR TO STMT ON STACK
0F6B  E3 ADD NEW ENTRY                XTHL        REMOVE IMPLIED JMP ADDR ØED6 FROM STACK, LEAVE IT IN HL
0F6C  F5                              PUSH  PSW   SAVE VARIABLE SIZE IN STACK   (since RST 4 uses ACC)
0F6D  D5                              PUSH  D     SAVE ADDR OF START OF 2ND TABLE (THIS IS WHERE VARIABLE IN DIM STMT WILL BE ADDED TO TABLE)
0F6E  11 11 0E                        LXI   D,X0E11  TEST IF STRING EVALUATOR (ØDCB) IS USING THE SYMBOL
0F71  E7                              RST   4       TABLE SEARCH ROUTINE AT ØE11- FIND VARIABLE IN SYMBOL
0F72  D1                              POP   D       TABLE + PUT VARIABLE VALUE IN HOLDING AREA
0F73  CA A6 0F                        JZ    X0FA6
0F76  F1                              POP   PSW   PUT VARIABLE IN ACC FROM STACK
0F77  E3                              XTHL        PUT ØED6 ON STACK, PUT PTR TO STMT IN HL
0F78  E5                              PUSH  H     PUT PTR TO STMT ON STACK
0F79  C5                              PUSH  B     PUT 2 CHAR'S OF VARIABLE NAME ON STACK (B IS 1ST C IS 2ND CHAR)
0F7A  4F                              MOV   C,A   SET UP BC   B IS ØØ   C IS VARIABLE-TYPE SIZE
0F7B  06 00                           MVI   B,H'00'
0F7D  C5                              PUSH  B     PUT SIZE INFO ON STACK (# OF BYTES FOR EACH ARRAY ELEMENT)
0F7E  03                              INX   B     ADD 3 TO SIZE (1 FOR DEFN-BYTE [2,3,4 or 8] & 2 FOR
0F7F  03                              INX   B                   VARIABLE NAME)
0F80  03                              INX   B
0F81  2A E5 03                        LHLD  X03E5 LOAD PTR TO END OF TABLE THAT FOLLOWS ARRAY TABLE
0F84  E5                              PUSH  H     SAVE THIS PTR ON STACK
0F85  09                              DAD   B     COMPUTE NEW ADDR FOR END OF 2ND TABLE
0F86  C1                              POP   B     PUT FORMER END OF 2ND TABLE PTR IN BC
0F87  E5                              PUSH  H     SAVE NEW END OF 2ND TABLE ADDR IN STACK
0F88  CD 78 04                        CALL  X0478 OPEN ENOUGH MEMORY TO MAKE ROOM FOR NEW ENTRY
0F8B  E1                              POP   H     STORE NEW END OF 2ND TABLE ADDR IN Ø3E5
0F8C  22 E5 03                        SHLD  X03E5
0F8F  60                              MOV   H,B   B,C ARE NEW ADDR FOR END OF SYMBOL TABLE (ACCOMPLISHED BY Ø478 ROUTINE)
0F90  69                              MOV   L,C
0F91  22 E3 03                        SHLD  X03E3 STORE NEW PTR VALUE
0F94  2B          X0F94               DCX   H     CLEAR ALL MEMORY LOC'S JUST OPENED
0F95  36 00                           MVI   M,H'00'  (ZERO VARIABLE VALUE JUST DEFINED)
0F97  E7                              RST   4
0F98  C2 94 0F                        JNZ   X0F94
0F9B  D1                              POP   D     PUT SIZE INFO IN REG E
0F9C  73                              MOV   M,E   STORE SIZE INFO AT 1ST BYTE JUST OPENED
0F9D  23                              INX   H     ADVANCE PTR TO TABLE
0F9E  D1                              POP   D     PUT ARRAY NAME IN DE (D IS 1ST E IS 2ND CHAR)
0F9F  73                              MOV   M,E   STORE 2ND CHAR OF NAME
0FA0  23                              INX   H     ADVANCE PTR
0FA1  72                              MOV   M,D   STORE 1ST CHAR OF NAME
0FA2  EB                              XCHG        DE POINT TO TABLE (BYTE CONTAINING 1ST CHAR)
0FA3  13          X0FA3               INX   D     ADVANCE PTR TO POINT TO VARIABLE VALUE
0FA4  E1                              POP   H     PUT PTR TO STMT IN HL
0FA5  C9                              RET         IMPLIED JMP TO ØED6 (NOT IF 'ERASE' OR 'LET')

0FA6  32 15 04    X0FA6               STA   X0415 STORE ØØ AT LOC Ø415 (IF FLOATING-PT VARIABLE NOT FOUND IN TABLE - VALUE IS ØØ)
0FA9  C1                              POP   B     FETCH VARIABLE SIZE FROM STACK (see ØF6C) - just removes 2 bytes from stack
0FAA  67          useless             MOV   H,A   PUT ØØ IN LOC Ø412 - Ø413 - IF INTEGER VARIABLE - FORCE 2 BYTES TO ØØ
0FAB  6F                              MOV   L,A
0FAC  22 12 04                        SHLD  X0412
0FAF  F7                              RST   6     TEST VARIABLE TYPE IN HOLDING AREA
0FB0  C2 1C 1A                        JNZ   X1A1C IF ARITHMETIC TYPE OF VARIABLE, PUT PTR TO STRING IN HL, THEN RETURN TO ØC5E
0FB3  21 47 04                        LXI   H,X0447  STORE ADDR OF NOP (PTR TO BYTE - SAYS STRING IS Ø IN LENGTH)
0FB6  22 12 04                        SHLD  X0412    AT Ø412 - Ø413
0FB9  E1                              POP   H     PUT PTR TO STRING IN HL (PUT ON STACK AT ØF6B)
0FBA  C9                              RET         RETURN TO ØC5E

0FBB  E5          X0FBB               PUSH  H     PUT VARIABLE TYPE (LOC Ø3A4) ON STACK
0FBC  2A A3 03                        LHLD  X03A3 HL STILL POINTS TO INSTRUCTION
0FBF  E3                              XTHL
```

SAVE Ø3A3 + Ø3A4 ON STACK SINCE STRING EVALUATOR USES THESE LOC'S

IF VARIABLE NOT IN TABLE PUT ØØ OR NULL STRING PTR IN HOLDING AREA, DON'T BOTHER CREATING THE VARIABLE IN THE SYMBOL TABLE

```
2540 SYMBOL TABLE
2581   BEFORE ØED6        AFTER ØED6
2582   PTR TO STMT        PTR TO STMT
2583
2584
2585 STMT WILL BE ADDED TO TABLE)
2586
2587
2588                                   C
2589                                   J&
2590
2591
2592
2593                                   E
2594                                   C
2595
2596
2597
2598
2599
2600
2601
2602
2603                                   A
2604
2605                                   M
2606
2607                                   -
2608
2609
2610
2611                                   +
2612                                   E
2613
2614                                   D,E
2615                                   B,C
2616
2617                                   #
2618                                   Q
2619
2620                                   Ø
2621
2622
2623
2624
2625                                   I
2625 - ENTRY NOT PUT IN TABLE YET.
2627
2628
2629
2630                                   -
2631
2632                                   B
2633                                   !G
2634                                   -
2635
2636                                   I
2637
2638 ARRAY TABLE SEARCH
2639
```





MOVE ARRAY TABLE TO MAKE ROOM FOR A NEW ENTRY AT END OF VARIABLE TABLE

```
0FC0  57                  MOV  D,A    CLEAR REG D (D IS DIMENSION CTR)
0FC1  D5            >0FC1 PUSH D      SAVE DIMENSION CTR ON STACK
0FC2  C5                  PUSH B      SAVE ARRAY NAME ON STACK
0FC3  CD E9 08            CALL X08E9  EVALUATE NEXT ARGUMENT  TO POSITIVE 2-BYTE INTEGER IN DE
0FC6  C1                  POP  B      PUT 2 CHAR OF ARRAY NAME BACK INTO BC
0FC7  F1                  POP  PSW    FETCH DIMENSION CTR INTO ACC FROM STACK.
0FC8  EB                  XCHG  } PUT PTR TO INSTRUCTION IN DE, PUT EVALUATED ARG ON STACK
0FC9  E3                  XTHL  } PUT VARIABLE TYPE IN HL
0FCA  E5                  PUSH H  PUT VARIABLE TYPE ON STACK AGAIN
0FCB  EB                  XCHG    PUT PTR TO INSTRUCTION IN HL
0FCC  3C                  INR  A  ADVANCE DIMENSION CTR
0FCD  57                  MOV  D,A  PUT DIMENSION CTR INTO ACC
0FCE  7E                  MOV  A,M  FETCH NEXT CHAR FROM INSTRUCTION STRING INTO ACC
0FCF  FE 2C               CPI  A',' } IF DELIMITER CHAR IS ',' ANOTHER DIMENSION ARG TO EVALUATE
0FD1  CA C1 0F            JZ   0FC1 }
0FD4  CF                  RST  1    } IF DELIMITER CHAR NOT A COMMA, THEN IT MUST BE A ')'
0FD5  29                  DAD  H    } PERFORM SYNTAX CHECK FOR ')'
0FD6  22 05 03            SHLD 0305 SAVE PTR TO INSTRUCTION IN TEMP STORAGE IN FLAG AREA
0FD9  E1                  POP  H    } FETCH ARRAY TYPE FROM STACK } 03A4 IS MODIFIED BY THE STRING EVALUATOR
0FDA  22 A3 03            SHLD 03A3 } STORE IN FLAG AREA
0FDD  1E 00               MVI  E,H'00' } CLEAR E, CLEARS FLAGS FOR POP PSW (SEE 1812)
0FDF  05                  PUSH D    } SAVE DIMENSION COUNT ON STACK
                                    MATCHED BY POP H AT 089F
ERASE 0FE0  11 F5 F5      LXI  D,F5F5 DUMMY PUSH H, PUSH PSW
(FROM 0F32) 0FE3  2A E3 03  LHLD 03E3 LOAD HL WITH ADDR OF FIRST BYTE IN ARRAY TABLE
            0FE6  3E 19     MVI  A,H'19 DUMMY  COMPUTE IN HL ADDR OF START OF NEXT ARRAY IN TABLE
      DAD D ->              XCHG  } LOAD DE WITH ADDR OF BYTE AFTER END OF ARRAY TABLE
            0FE8  EB        LHLD 03E5 } HL IS PTR USED TO SEARCH ARRAY TABLE
            0FE9  2A E5 03  XCHG  }
            0FEC  E7        RST  4    TEST (HL) AGAINST (DE)
            0FED  3A A4 03  LDA  03A4 LOAD ACC WITH VARIABLE TYPE
            0FF0  CA 1F 10  JZ   101F IF (HL) = (DE), NO ENTRIES IN TABLE OR ARRAY NOT FOUND
TABLE EMPTY                           IN TABLE
OR NOT FOUND 0FF3  BE       CMP  M    TEST VAR TYPE OF ARRAY IN TABLE TO SEE IF MATCH TYPE LOOKING FOR
IN TABLE     0FF4  23       INX  H    ADVANCE PTR TO ARRAY TABLE
NO MATCH     0FF5  C2 02 10 JNZ  1002 JMP IF NO MATCH, ARRAY IN TABLE NOT THE ONE WE WANT
             0FF8  7E       MOV  A,M  FETCH 2ND CHAR OF ARRAY IN TABLE INTO ACC
             0FF9  B9       CMP  C    COMPARE WITH 2ND CHAR OF ARRAY IN INSTRUCTION
             0FFA  23       INX  H    ADVANCE PTR TO ARRAY TABLE
NO MATCH     0FFB  C2 03 10 JNZ  1003 JMP IF NO MATCH
             0FFE  7E       MOV  A,M  FETCH 1ST CHAR OF ARRAY IN TABLE INTO ACC
             0FFF  B8       CMP  B    COMPARE WITH 1ST CHAR OF ARRAY IN INSTRUCTION
             1000  3E 23    MVI  A,A'#' DUMMY
INX H ->     1001  23       INX  H    ADVANCE PTR TO ARRAY TABLE
X1003 ->     1002  5E       MOV  E,M  } IF NO MATCH, DE CONTAINS # OF BYTES TO ADD TO HL PTR
             1003  23       INX  H    } SKIP OVER THIS ARRAY (NOTICE DAD D AT 0FE7)
             1004  56       MOV  D,M  } OTHERWISE, HL POINTS TO BYTE CONTAINING # OF DIMENSIONS
             1005  23       INX  H    }
             1006  C2 E7 0F JNZ  0FE7 LOOP IF ARRAY IN TABLE NOT THE ARRAY TO BE FOUND
             1009  3A A3 03  LDA  03A3 } TEST 03A3, IF HERE, WE ARE PROCESSING A. DIM STMT, + WE JUST
             100C  B7       ORA  A    } FOUND THE ARRAY IN THE TABLE. DIM FORCES 03A4 TO AF AT 0E1F
'DIM' <-     100D  C2 B2 04 JNZ  04B2 PRINT ERROR MESSAGE "REDIMENSIONED ARRAY" IF 03A3 NOT B200
             1010  F1       POP  PSW  FETCH DIMENSION COUNT INTO ACC, DIM FORCED ALL FLAGS TO 00 AT 0FDD
'ERASE' <-   1011  CA 92 08 JZ   0892 JMP IF ERASE COMMAND (ZERO FLAG SET WHEN I2 TO 0FE1 FROM 0F32)
             1014  96       SUB  M    TEST IF # OF DIMEN. OF ARRAY IN TABLE MATCHES # OF DIMEN OF ARRAY IN
'LET' <-     1015  CA 7A 10 JZ   107A JMP IF # OF DIMENSIONS CRITERIA HAS BEEN MET
             1018  1E 09    X101A MVI  E,H'09 } PRINT
             101A  C3 B7 04 JMP  04B7 } ERROR MESSAGE "SUBSCRIPT OUT OF RANGE"
             101D  77       X101F MOV  M,A    STORE VAR TYPE AT END OF ARRAY TABLE (NEW ENTRY TO TABLE)
             101E  23              INX  H    ADVANCE PTR TO TABLE
             101F  [illegible]     [illegible] SET UP DE FOR MULTIPLY LOOP THAT [illegible] BYTES ARE NEEDED
```

H	2640	
U	2641	
E	2642	
M DE	2643	HI MEM
A	2644	
	2645	
	2646	
	2647	
	2648	
	2649	
<	2650	
H	2651	
	2652	
	2653	
JA	2654	
'C	2655	LO MEM
)	2656	
	2657	
	2658	
"#	2659	
	2660	
U	2661	
	2662	
*	2663	
TABLE	2664	LO-MEM
	2665	
*	2666	
	2667	LO
	2668	HI
:3	2669	
IN TABLE	2670	
FOR	2671	LO
#	2672	
0	2673	HI
	2674	LO
9	2675	HI
#	2676	
E	2677	
	2678	
8	2679	
>0	2680	LO
0	2681	HI
TO	2682	
#	2683	
IONS	2684	
#	2685	
8	2686	
IA	2687	
F	2688	
	2689	
	2690	
	2691	
INSTR.	2692	HI-MEM
J	2693	
	2694	
C7	2695	
	2696	
#	2697	
-	2698	
	2699	

STACK

	1ST ARG
	2ND ARG
	3RD ARG
	Nth ARG
	ARRAY TYPE (02 03 04 05)

STACK CONTENTS AFTER FALLING OUT OF LOOP AT 0FD1

FORMAT OF ARRAY IN ARRAY TABLE

	VAR. TYPE	= S
	2ND CHAR	
	1ST CHAR	
LO	# OF BYTES	= 3 + N*S ?
HI	IN ARRAY	
	# OF DIMEN.	
LO	# OF ELEMENTS	= K_N
HI	ARG_N + 1	
LO	# ELEMENTS	= K_{N-1}
HI	ARG_{N-1} + 1	
		$N = K_1 * \ldots * K_{N-1} * K_N$
LO	# ELEMENTS	= K_1
HI	ARG1 + 1	
	ARRAY ELEMENTS	

ADD NEW ARRAY TO ARRAY TABLE

17

```
1024  F1                    POP   PSW FETCH DIMENSION COUNT INTO ACC, D      FORCED ALL FLAGS TO ØØ AT ØFDØ   2700
1025  CA EE 08  <'ERASE'    JZ    X08EE JMP IF ERASE COMMAND (ZERO FLAG SET WHEN JZ TO ØFE1 FROM ØF32)    2701 PRINT "ILLEGAL FUNCTION CALL"
1028  71                    MOV   M,C  } STORE 2 CHAR'S OF ARRAY NAME INTO TABLE, 2ND CHAR, THEN 1ST       2702
1029  23                    INX   H    }                                                    #              2703
102A  70                    MOV   M,B  }                                                                   2704
102B  23                    INX   H    }                                                    #              2705
102C  4F                    MOV   C,A  PUT # OF DIMENSIONS IN ACC.                          O              2706 REASON - NEED 2 BYTES TO STORE
102D  CD 87 04              CALL  X0487 TEST IF ENOUGH MEMORY FOR (2X # OF DIMENSIONS) AVAILABLE            2707 EACH (ARG+1) = # OF ELEMENTS IN
1030  23                    INX   H  } ADVANCE PTR TO ARRAY TABLE TO POINT TO BYTE WHERE # OF               2708 THAT DIMENSION
1031  23                    INX   H  } DIMENSIONS WILL BE STORED                                           2709
1032  22 CD 03              SHLD  X03CD STORE ADDR OF BYTE CONTAINING # OF DIMENSIONS IN NEW ENTRY         2710
1035  71                    MOV   M,C  } STORE BYTE - # OF DIMENSIONS IN ARRAY TABLE                       2711
1036  23                    INX   H    }                                                    #              2712
1037  3A A3 03              LDA   X03A3  } LOAD & TEST BRANCHING FLAG STORED AT Ø3A3                       2713
103A  17                    RAL          } (Ø3A3) = AF => DIM    (Ø3A3) = ØØ => LET                        2714
103B  79                    MOV   A,C  PUT # OF DIMENSIONS IN ACC                                          2715
                                                                                                  MULTIPLY LOOP
                                                                                                  CUMULATIVE RESULT IN DE
                                                                                                  MULTIPLIER IN BC
                                                                                                  (DE) = (DE) * (BC)
                                                                                                  2-BYTE INTEGER MULTIPLY
103C  01 09 00  >X103C      LXI   B,X0009 SET DEFAULT SIZE TO 11 DECIMAL IF 'LET' (USING AN ARRAY IN LET   2716 PRIOR TO A 'DIM' DEFAULTS EACH DIM TO
                                                                                                                11 DECIMAL
103F  D2 44 10              JNC   X1044 JMP IF PROCESSING A 'LET' INSTRUCTION                  RD          2717
1042  C1                    POP   B  FETCH AN ARGUMENT FROM STACK (LAST ARG FIRST)             A           2718
1043  03                    INX   B  ADD 1 TO ARG (Ø^th ELEMENT INCLUDED)                                  2719
1044  71                  >X1044  MOV   M,C  } STORE 2-BYTES CONTAINING # OF ELEMENTS IN DIMENSION        2720
1045  23                    INX   H    }                                  BEING PROCESSED  #              2721
1046  70                    MOV   M,B  }                                                                   2722
1047  23                    INX   H    }                                                    #              2723
1048  F5                    PUSH  PSW SAVE DIMENSION (ARGUMENT) CTR ON STACK & FLAGS                       2724
1049  CD 32 10              CALL  X1032 MULTIPLY # OF ELEMENTS IN ARRAY (FORM PRODUCT [VAR. TYPE] X ARG1   M2 X  2725 ARG2 X . . . . X ARGN)
104C  F1                    POP   PSW FETCH DIMENSION CTR FROM STACK + FLAGS                               2726
104D  3D                    DCR   A  DECREMENT DIMENSION CTR                                 =           2727
104E  C2 3C 10              JNZ   X103C IF NOT ZERO, LOOP TO PROCESS ANOTHER ARGUMENT          B<          2728
1051  F5                    PUSH  PSW SAVE ACC (= ØØ) & CARRY FLAG ON STACK                                2729
1052  42                    MOV   B,D  } SAVE # OF BYTES TO STORE ARRAY ELEMENTS IN BC                     2730
1053  4B                    MOV   C,E  }                                                     K             2731
1054  EB                    XCHG  PUT PTR-TO 1ST BYTE OF 1ST ELEMENT IN DE, PUT # OF BYTES IN HL           2732
1055  19                    DAD   D  COMPUTE ADDR OF BYTE AFTER END OF ARRAY TABLE WITH NEW ARRAY INSTALLED 2733
1056  DA 1A 10              JC    X101A PRINT ERROR MESSAGE "SUBSCRIPT OUT OF RANGE" IF MULTIPLY PRODUCED  2734 AN OVERFLOW
1059  CD 94 04              CALL  X0494 TEST FOR AVAILABLE MEMORY SPACE - ERROR MESSAGE IF NOT ENOUGH      2735
105C  22 E5 03              SHLD  X03E5 STORE ADDR OF BYTE AFTER END OF ARRAY TABLE             -          2736
105F  2B                  >X105F  DCX   H  BACK-UP PTR TO ELEMENTS IN ARRAY TABLE               +          2737
1060  36 00                 MVI   M,H'00' CLEAR ALL BYTES OF EVERY ELEMENT TO Ø                6          2738
1062  E7                    RST   4  TEST IF (HL) = (DE)                                                   2739
1063  C2 5F 10              JNZ   X105F CONTINUE TO CLEAR ELEMENTS IN ARRAY UNTIL (HL) = (DE)  B_         2740
1066  03                    INX   B  ADJUST BC SO THAT # IS # OF BYTES TO STORE ELEMENTS +1 (= # OF BYTES FOR 2741 ELEMENTS + 1 FOR #-OF DIMENSIONS BYTE)
1067  57                    MOV   D,A CLEAR D REG (ACC IS ØØ WHEN FALLING OUT OF LOOP AT 1Ø4E)  W        2742
1068  2A CD 03              LHLD  X03CD LOAD HL WITH ADDR OF BYTE CONTAINING # OF DIMENSIONS    *M         2743
106B  5E                    MOV   E,M PUT #-OF-DIMENSIONS BYTE IN E, D IS ØØ                   ^          2744
106C  EB                    XCHG  PUT PTR IN DE, # OF DIMENSIONS IN HL                                     2745
106D  29                    DAD   H MULTIPLY #-OF-DIMENSIONS BY 2                              )          2746
106E  09                    DAD   B ADD #-OF-BYTES TO STORE ALL ELEMENTS IN ARRAY + #-OF-DIMEN. BYTE       2747
106F  EB                    XCHG  PUT PTR BACK IN HL, SUM OF # OF BYTES TOTAL TO STORE ARRAY               2748
1070  2B                    DCX   H  } BACK-UP PTR TO POINT TO "CHAIN ADDR" OF ARRAY           +          2749
1071  2B                    DCX   H  }                                                         +          2750
1072  73                    MOV   M,E  } STORE 2 BYTES CONTAINING VALUE TO TELL # OF BYTES                 2751
1073  23                    INX   H    } NEEDED TO STORE: 1 BYTE FOR # OF DIMEN                #          2752.
1074  72                    MOV   M,D  }                + 2 BYTES FOR EACH DIMEN - # OF ELEMENTS IN EACH 2753 DIMEN
1075  23                    INX   H    }                + # OF BYTES FOR ALL ELEMENTS           #          2754
1076  F1                    POP   PSW FETCH ACC (= ØØ) & CARRY FLAG FROM STACK                             2755
1077  DA AC 10  <'DIM'      JC    X10AC IF CARRY IS SET (SEE 1Ø37), END OF PROCESSING 1 ARRAY FOR DIM,  Z.  2756 EVENTUALLY IMPLIED JMP TO ØED6
                                                                                                    C       2757
107A  47                  X107A   MOV   B,A  } SET BC TO ØØØØ                                    O          2758 PROCESSING LET WITH AN
107B  4F                    MOV   C,A  }                                                                   2759 ARRAY ELEMENT ON LEFT
107C  7E                    MOV   A,M FETCH BYTE CONTAINING # OF DIMENSIONS INTO ACC                           OF '='
```

```
1070  23                  INX  H ADVANCE PTR TO ARRAY TABLE (PTS TO LO-BYTE OF # ELEMENTS    #     2760
107E  16 E1               MVI  D,H'E1 *DUMMY POP H                                          2761
1080  5E                  MOV  E,M  } LOAD DE WITH 2 BYTE VALUE = # OF ELEMENTS IN DIMENSION BEING  2762 PROCESSED
1081  23                  INX  H    } ADVANCE POINTER TO ARRAY TABLE                        #     2763
1082  56                  MOV  D,M  }                                                       V     2764
1083  23                  INX  H    }                                                       #     2765
1084  E3                  XTHL PUT 1 ARG (# INDICATING WHICH ELEMENT) IN HL, PUT PTR TO ARRAY TABLE ON  2766 STACK
1085  F5                  PUSH PSW SAVE DIMENSION COUNTER ON STACK                          2767
1086  E7                  RST  4 TEST ARGUMENT SIZE AGAINST # OF ELEMENTS IN THAT ARRAY AS DIMENSIONED  2768
1087  D2 1A 10            JNC  X101A IF ARG ≥ ARRAY SIZE (IN THAT DIMENSION) PRINT "SUBSCRIPT OUT OF RANGE"  R  2769
108A  CD 32 10            CALL X1032 MULTIPLY (INTEGER) DIMENSION SIZE * ACCUMULATED VALUE  M2  2770 } FORM PRODUCT-SUM TO COMPUTE 1
108D  19                  DAD  D  ADD ARGUMENT TO ACCUMULATED PRODUCT (DE)                  2771 } DIMENSIONAL OFFSET FROM AN N-
108E  F1                  POP  PSW FETCH (HL) DIMENSION CTR FROM STACK                      2772 } DIMENSIONAL ARRAY
108F  3D                  DCR  A  DECREMENT DIMENSION CTR                                   =     2773 } OFFSET = ((ARG3) * DIM2 + ARG2) * SIZE1
1090  44                  MOV  B,H  } PUT ACCUMULATED VALUE IN BC                           D     2774 }          + ARG1
1091  4D                  MOV  C,L  }                                                       M     2775 } SHOWN FOR 3 DIMENSIONAL ARRAY
1092  C2 7F 10            JNZ  X107F LOOP UNTIL ALL ARGUMENTS HAVE BEEN PROCESSED           B     2776
1095  3A A4 03            LDA  X03A4 PUT VARIABLE TYPE IN ACC                               :$    2777
1098  44                  MOV  B,H  } PUT OFFSET FOR ARRAY ELEMENT IN BC                    D     2778
1099  4D                  MOV  C,L  }                                                       M     2779
109A  29                  DAD  H  MULTIPLY OFFSET BY 2 (INTEGER TYPE VALUE - 2 BYTES PER ELEMENT)  2780
109B  D6 04               SUI  H'04' TEST VARIABLE VALUE IN ACC                             V     2781
109D  DA A5 10            JC   X10A5 JMP IF INTEGER TYPE VALUE                              Z%    2782
10A0  29                  DAD  H  MULTIPLY OFFSET BY 2 (SINGLE-PRECISION VALUE - 4 BYTES PER ELEMENT)  2783
10A1  CA A9 10            JZ   X10A9 JMP IF SINGLE-PREC VALUE                               J)    2784
10A4  29                  DAD  H  MULTIPLY OFFSET BY 2 (DOUBLE-PRECISION VALUE - 8 BYTES PER ELEMENT)  2785
10A5  E2 A9 10            JPO  X10A9 JMP IF INTEGER OR DOUBLE-PRECISION                     J)    2786
10A8  09                  DAD  B  ADD OFFSET TO 2* OFFSET. ALREADY IN HL (STRING VALUE - 3 BYTES PER ELEMENT)  2787
10A9  C1                  POP  B  FETCH PTR TO ARRAY TABLE FROM STACK                       A     2788
10AA  09                  DAD  B  COMPUTE ADDR WHERE VALUE OF DESIRED ELEMENT IS LOCATED IN ARRAY TABLE  2789
10AB  EB                  XCHG PUT ADDR WHERE ARRAY ELEMENT LOCATED IN DE (PTR TO VAR VALUE REGION)  2790
10AC  2A C5 03      X10AC LHLD X03C5 RESTORE PTR TO INSTRUCTION STRING FROM TEMP STORAGE IN  *U    2791
10AF  C9                  RET       FLAG AREA                                             I     2792
```

INT & STRING, SINGLE PRECISION, X10A5, INT & DBL, X10A9 — branch targets for the JC, JZ, JPO and RST 4 instructions above.

```
10B0  2A E5 03            LHLD X03E5 PUT ADDR OF FIRST BYTE AVAILABLE AFTER END OF ARRAY TABLE IN HL*  2793  FRE
10B3  EB                  XCHG       SAVE THIS ADDR IN DE                                  2794
10B4  21 00 00            LXI  H,X0000 } PUT CURRENT STACK PTR ADDR IN HL                  !     2795
10B7  39                  DAD  SP      }                                                   9     2796
10B8  F7                  RST  6   TEST VALUE TYPE                                          2797
10B9  C2 C9 10            JNZ  X10C9 JMP IF NOT A STRING VALUE (COMPUTE AMOUNT OF ARITHMETIC  BI  2798 SPACE AVAILABLE)
10BC  CD 7A 13            CALL X137A CANCEL ARG STRING                                      M     2799
10BF  CD 30 12            CALL X1230 PURGE                                                  M     2800
10C2  2A 00 03            LHLD X0300 LOAD HL WITH ADDR OF "BOTTOM" OF STACK                 *}    2801
10C5  EB                  XCHG  PUT STACK START ADDR IN DE                                  2802
10C6  2A C8 03            LHLD X03C8 LOAD HL WITH ADDR OF NXT BYTE AVAILABLE IN STRING AREA  *K  2803
10C9  7D            X10C9 MOV  A,L } SUBTRACT ADDR IN DE FROM ADDR IN HL                    2804
10CA  93                  SUB  E    } RESULT IS AMOUNT OF MEMORY AVAILABLE                  2805
10CB  6F                  MOV  L,A  }                                                       2806
10CC  7C                  MOV  A,H  }                                                       2807
10CD  9A                  SBB  D    }                                                       2808
10CE  C3 D6 10            JMP  X10D6 TREAT # IN HL AS AN INTEGER, STORE IN HOLDING AREA     CV    2809
10D1  3A 27 00            LDA  X0027 LOAD PRINT POSITION CTR INTO ACC, TREAT AS INTEGER, PUT IT IN HL!*  2810  POS
10D4  6F            X10D4 MOV  L,A PUT ACC IN L REG } PUT BYTE IN L, CLEAR H - RESULT IS INTEGER VALUE IN  2811
10D5  AF                  XRA  A   CLEAR ACC          }                                 / HL  2812
10D6  67            X10D6 MOV  H,A PUT ACC IN H REG  }                                      2813
10D7  C3 20 1C            JMP  X1C20 STORE INTEGER VALUE AT Ø412 & Ø413, SET VALUE LOC (Ø3A4) TO Ø2  C-  2814
10DA  CD C1 11            CALL X11C1 PERFORM SYNTAX CHECK FOR 'FN' TOKEN, FIND OR CREATE A VARIABLE IN THE SYMBOL TABLE  MA  2815 (NAME OF FCN WITH D7 OF 1ST CHAR SET)  DEF
10DD  CD B3 11            CALL X11B3 TEST IF DEF COMMAND BEING USED IN DIRECT MODE - ILLEGAL - PRINT ERROR MESSAGE  M3  2816
10E0  EB                  XCHG PUT PTR TO VARIABLE VALUE REGION IN HL (see ØFA3), PUT PTR TO '(' IN DEF STMT IN DE  2817
10E1  73                  MOV  M,E } PUT PTR TO DEFINITION STRING IN VARIABLE VALUE REGION  2818
10E2  23                  INX  H   }                                                        #     2819  DEFINE USER FUNCTION
```

EVALUATE USER FUNCTION

```
10E3  72                      MOV  M,D   } PUT PTR TO DEFINITION STRING IN VARIABLE VALUE REGION
10E4  EB                      XCHG       PUT PTR TO STRING IN HL
10E5  C3 8C 09                JMP  X098C PUSH STMT PTR PAST REST OF INSTRUCTION - SCAN TIL NULL OR COLON FOUND
10E8  CC C1 11    X10E8       CALL X11C1 PERFORM SYNTAX CHECK FOR 'FN' TOKEN, FIND OR CREATE A VARIABLE IN THE SYMBOL TABLE
10EB  3A A4 03                LDA  X03A4 } TEST FLAG AREA LOC. FOR TYPE OF VARIABLE (PRECISION OF FCN-
10EE  B7                      ORA  A     } SET FLAGS & PUT TYPE-CODE & FLAGS ON THE STACK
10EF  F5                      PUSH PSW   } CLEAR CARRY BIT FOR TEST AT 1165
10F0  22 D5 03                SHLD X03D5 SAVE PTR TO STRING OF FCN TO BE EVALUATED (SHOULD POINT TO '(')
10F3  EB                      XCHG       PUT PTR TO FCN-NAME VARIABLE VALUE REGION IN HL (see ØFA3)
10F4  7E                      MOV  A,M   } PUT PTR TO DEFINITION STRING IN HL (see 10E1-10E3)
10F5  23                      INX  H     }
10F6  66                      MOV  H,M   }
10F7  6F                      MOV  L,A   }
10F8  B4                      ORA  H     } TEST HL
10F9  CA D5 04                JZ   X04D5 } IF HL IS ØØØØ, THEN FCN NOT YET DEFINED, PRINT ERROR
10FC  7E                      MOV  A,M   FETCH A CHAR FROM DEFINITION STRING
10FD  FE 28                   CPI  A'('  } TEST CHAR & JMP IF CHAR IS NOT A '('
10FF  C2 51 11                JNZ  X1151 }
1102  D7                      RST  2     SCAN TO NEXT NON-SPACE CHAR IN DEFINITION STRING
1103  22 CD 03                SHLD X03CD SAVE PTR TO 1ST CHAR OF VARIABLE-ARG NAME AT ØJCD (IN DEFINITION STRING)
1106  01 CF 2C    RST1 DB',' LXI  B,X2CCFDummy
1109  0E 04                   MVI  C,H'04' } TEST IF ENOUGH SPACE BETWEEN END OF ARRAY TABLE & STACK POINTER
110B  CD 87 04                CALL X0487   } POSITION FOR 8-BYTES
110E  3E 80                   MVI  A,H'80' } SET Ø3D1 TO 8Ø TO INDICATE DEF IS USING SYMBOL TABLE SEARCH ROUTINE
1110  32 D1 03                STA  X03D1   }
1113  CD E0 0E                CALL X0EE0 FIND VARIABLE-ARG IN THE SYMBOL TABLE
1116  EB                      XCHG       PUT PTR TO VARIABLE-ARG VALUE REGION IN HL (see ØFA3), PUT PTR TO NXT NON-SPACE CHAR IN DE (SHOULD POINT TO ')' IN DEFINITION STRING
1117  37                      STC        SET CARRY - FORCES RETURN TO 1125 FROM ØCBE ROUTINE (see ØCE6)
1118  C3 BE 0C                JMP  X0CBE MOVE VALUE FROM VARIABLE-ARG VALUE REGION ONTO STACK
111B  D2 8A 1C    X111B       JNC  X1C8A IF CARRY NOT SET, MISMATCH IN VARIABLES - (see ØC98 FOR STRING VARIABLE BRANCH) STRING TYPE VARIABLE
111E  D5                      PUSH D     SAVE DE ON STACK - PTR TO DEFINITION STRING
111F  EB                      XCHG       PUT PTR TO VARIABLE-ARG VALUE REGION IN DE - 1222 WILL SAVE IT ON STACK, PUT IT IN HL WHEN DONE
1120  CD 22 12                CALL X1222 DE PTS TO IDENTITY GROUP LOC, PUT IDENTITY GROUP IN STRING FORMULA TABLE
1123  D1                      POP  D     RESTORE DE FROM STACK - PTR TO DEFINITION STRING
1124  AF                      XRA  A     CLEAR ACC = (TYPE-3) = Ø FOR STRING VALUE
1125  F5          X1125       PUSH H     SAVE PTR TO LAST BYTE OF VARIABLE-ARG VALUE REGION ON STACK
1126  F5                      PUSH PSW   SAVE (TYPE-3) OF VALUE ON STACK & FLAGS JZ-STRING JM-INTEGER JPO-SNG PREC
1127  EB                      XCHG       PUT PTR TO DEFINITION STRING IN HL (see 1116)
1128  7E                      MOV  A,M   FETCH A CHAR FROM DEFINITION STRING
1129  FE 29                   CPI  A')'  } TEST CHAR - IF CHAR IS NOT A ')', PERFORM A SYNTAX CHECK FOR ')'
112B  C2 37 11                JNZ  X1107 } REPEAT THE ABOVE PROCEDURE FOR ANOTHER VARIABLE-ARG IN THE
112E  2A D5 03                LHLD X03D5 LOAD HL WITH PTR TO FCN-STRING TO BE EVALUATED
1131  CF                      RST  1     } PERFORM SYNTAX CHECK FOR '('
1132  28                      DATA A'('  }
1133  E5                      PUSH H     PUT PTR TO FCN-STRING TO BE EVALUATED ON STACK
1134  2A CD 03                LHLD X03CD LOAD HL WITH PTR TO 1ST CHAR OF VARIABLE-ARG NAME AT ØJCD (see 1103)
1137  CD E0 0E    X1137       CALL X0EE0 FIND VARIABLE-ARG IN THE SYMBOL TABLE - HL PTS TO DEFN-STRING
113A  E3                      XTHL       PUT PTR TO FCN-STRING TO EVALUATE IN HL, PUT PTR TO DEFN-STRING ON STACK
113B  CD B5 09                CALL X09B5 EVALUATE ARG IN FCN-STRING - STORE IT IN VARIABLE-ARG VALUE REGION
113E  7E                      MOV  A,M   FETCH A CHAR FROM FCN-STRING TO BE EVALUATED
113F  FE 29                   CPI  A')'  } TEST CHAR - IF CHAR IS ')' IN FCN STRING, NO MORE ARGS TO BE ASSIGNED
1141  CA 4C 11                JZ   X114C } TO VARIABLES IN THE DEFINITION-STRING
1144  CF                      RST  1     } PERFORM SYNTAX CHECK FOR ',' IN FCN-STRING TO BE EVALUATED
1145  2C          DB ',' INR  L          }
1146  E3                      XTHL       PUT PTR TO DEFN-STRING IN HL, PUT PTR TO FCN-STRING TO BE EVALUATED ON STACK
1147  CF                      RST  1     } PERFORM SYNTAX CHECK FOR ',' IN DEFN-STRING
1148  2C                      INR  L     }
1149  C3 37 11                JMP  X1137 LOOP UNTIL ALL ARG-VALUES HAVE BEEN EVALUATED & PUT INTO THE
114C  E1                      POP  H     APPROPRIATE VARIABLE VALUE REGIONS
```

```
114E CF          RST  1   } PERFORM SYNTAX CHECK FOR ')' IN DE. . - STRING                    O   2880
114F 29          DB ')'   } THIS GUARANTEES THAT THE # OF ARGS IN EACH STRING MUST MATCH     )   2881
1150 3E 05       PUSH D MVI A,H'05' DUMMY                                                    >U  2882
1152 CF          RST  1   } PERFORM SYNTAX CHECK FOR '=' TOKEN IN DEFN STRING                O   2883
1153 88          DB '88'  }                                                                  8   2884
1154 CC 52 0C    CALL X0C52 EVALUATE DEFN-STRING USING ARGS STUFFED INTO VARIABLE-ARGS OF FCN MR  2885
1157 2B          DCX  H   BACK-UP PTR TO DEFN-STRING                                         +   2886 (see 1137-1149)
1158 D7          RST  2   SCAN TO NEXT NON-SPACE CHAR IN DEFN-STRING - TEST THIS CHAR        W   2887
1159 C2 A9 04    JNZ  X04A9 IF DELIMITER IN DEFN-STRING IS NOT A NULL OR COLON- PRINT ERROR  B)  2888
115C D1          POP  D   PUT PTR TO FCN-STRING TO BE EVALUATED IN DE                        Q   2889 MESSAGE 'SYNTAX ERROR'
115D F7          RST  6   TEST VALUE-TYPE CODE OF EVALUATED FCN                                  2890
115F CA 9F 11    JZ   X119F JMP IF FCN IS STRING TYPE- PUT IDENTITY GROUP FOR FCN VALUE AT END OF STRING  2891 FORMULA TABLE - WHETHER OR NOT IT'S ALREADY THERE
1161 F1       X1161 POP  PSW FETCH (TYPE-CODE -3) FLAGS FROM STACK (see 1126)                    2892
1162 CA A3 11    JZ   X11A3 JMP IF VARIABLE VALUE TO BE RESTORED IS STRING TYPE              J#  2893   INVERSE OF 1106-112B
1165 D2 85 11    JNC  X1185 JMP IF CARRY NOT SET - DONE RESTORING VARIABLE VALUES - POPPED BYTE PUSHED BY 10EF  R  2894
1168 E1          POP  H   FETCH PTR TO LAST BYTE OF VARIABLE-ARG VALUE REGION FROM STACK (see 1125)  2895
1169 C1          POP  B   FETCH 2 BYTES OF VARIABLE VALUE FROM STACK                         A   2896
116A 70          MOV  M,B } STORE THESE 2 BYTES BACK INTO THE VARIABLE VALUE REGION IN THE    +   2897
116B 2B          DCX  H   } SYMBOL TABLE                                                         2898
116C 71          MOV  M,C }                                                                      2899
116D FA 61 11    JM   X1161 JMP IF VARIABLE IS INTEGER TYPE - ONLY 2 BYTES TO RESTORE            2900
1170 2B          DCX  H   }                                                                   +   2901
1171 C1          POP  B   } FETCH 2 MORE BYTES FROM THE STACK + STORE THEM IN                A   2902
1172 70          MOV  M,B } THE VARIABLE VALUE REGION.                                           2903
1173 2B          DCX  H   }                                                                   +   2904
1174 71          MOV  M,C }                                                                      2905
1175 E2 61 11    JPO  X1161 JMP IF VARIABLE IS SNG-PREC TYPE - ONLY 4 BYTES TO RESTORE           2906
1178 2B          DCX  H   }                                                                   +   2907
1179 C1          POP  B   } FETCH 4 MORE BYTES FROM THE STACK + STORE THEM                   A   2908
117A 70          MOV  M,B } IN THE VARIABLE VALUE REGION                                         2909
117B 2B          DCX  H   }                                                                   +   2910
117C 71          MOV  M,C }                                                                      2911
117D 2B          DCX  H   }                                                                   +   2912
117E C1          POP  B   }                                                                   A   2913
117F 70          MOV  M,B }                                                                      2914
1180 2B          DCX  H   }                                                                   +   2915
1181 71          MOV  M,C }                                                                      2916
1182 C3 61 11    JMP  X1161 LOOP UNTIL ALL VARIABLES HAVE HAD THEIR VALUES RESTORED          C   2917
1185 D5       X1185 PUSH D PUT PTR TO FCN-STRING TO BE EVALUATED ON STACK .                   U   2918
1186 F5          PUSH PSW PUT TYPE-CODE FOR FCN PRECISION ON STACK (see 10EB-10EF)               2919
1187 F7          RST  6   TEST VALUE TYPE-CODE OF EVALUATED FCN (VALUE IN HOLDING AREA)          2920
1188 11 CB 03    LXI  D,X03CB LOAD DE WITH ADDR OF LAST 3 LOC'S OF STRING FORMULA TABLE      H   2921
118B CC 22 12    CZ   X1222 IF FCN-VALUE IS STRING TYPE, PUT IDENTITY GROUP OF FCN-VALUE IN STRING FORMULA TABLE (see 119F-11AF)  2922
118E F1          POP  PSW FETCH TYPE-CODE FOR FCN PRECISION FROM STACK                           2923
-------------------------------------------------------------------------------------------------------------------------------
1190 118F FE E5  DUMMY PUSH H  CPI  H'E5' LET - PUT PTR TO VAR VALUE REGION ON STACK  { INT = 2 CINT
1191 E6 07       ANI  H'07' CLEAR UPPER 5 BITS OF TYPE CODE IN ACC  { CHAR = 3 TEST + RETURN
1193 21 F1 01    LXI  H,X01F1 PUT ADDR 01F1 IN HL  { SNG = 4 CSGN
1196 4F          MOV  C,A   } ADD 2X TYPE CODE TO ADDR IN HL                                 O   2927 TYPE-CODE IN ACC
1197 06 00       MVI  B,H'00' } FETCH ADDR OF SUBR FROM TABLE, JMP TO THAT SUBR                  2928
1199 09          DAD  B     }                                                                    2929
119A CD 59 0E    CALL X0E59 }                                                                MY  2930
119D D1          POP  H   PUT PTR TO VAR VALUE REGION ON STACK                                  2931
119E C9          RET  DONE                                                                   I   2932
-------------------------------------------------------------------------------------------------------------------------------
119F 21 CB 03 X119F LXI  H,X03CB } PUT THIS END OF TABLE ADDR ON STACK TO MOVE FCN-VALUE IDENTITY  !H  2933
11A2 E5          PUSH H     } GROUP AT LAST 3 BYTES OF STRING FORMULA TABLE                      2934
11A3 CD 96 13 X11A3 CALL X1396 REMOVE IDENTITY GROUP FROM STRING FORMULA TABLE (BACK UP THE PTR TO TABLE BUT DON'T ADJUST FLAG AREA PTR (03A8) YET)  M  2935
11A6 7E          MOV  A,M   PUT CHAR COUNT BYTE OF REMOVED GROUP IN ACC                         2936
11A7 22 A8 03    SHLD X03A8 ADJUST FLAG AREA PTR TO NEW VALUE (REMOVES GROUP FROM TABLE)     "(  2937
11AA E1          POP  H   PUT ADDR WHERE IDENTITY GROUP SHOULD BE STORED IN HL (FCN VALUE IDENT GROUP AT END OF TABLE, OR 1ST BYTE ADDR OF VARIABLE VALUE REGION - see 1125 coming from 1128 no change of HL in 1222)  2938
11AB 77          MOV  M,A   } PUT CHAR COUNT BYTE INTO VARIABLE (1ST BYTE) VALUE REGION          2939
```

```
11AC  23          INX  H    } PUT PTR ADDR OF IDENTITY GROUP INTO 2ND + 3RD BYTES OF
11AD  71          MOV  M,C  } VARIABLE VALUE REGION
11AE  23          INX  H
11AF  70          MOV  M,B
1180  C3 61 11    JMP  X1161 LOOP UNTIL ALL VARIABLES HAVE HAD THEIR VALUES RESTORED
-------------------------------------------------------------------------------------
1183  E5    X1183 PUSH H SAVE HL ON STACK
1184  2A 07 03    LHLD X0307 LOAD LINE # OF CURRENT INSTRUCTION FROM FLAG AREA
1187  23          INX  H    } ACC IS ØØ ONLY IF HL =FFFF
1188  7C          MOV  A,H  }
1189  B5          ORA  L    }
118A  E1          POP  H RESTORE HL FROM STACK
118B  C0          RNZ  RETURN IF PROGRAM IS RUNNING, OR HAS BEEN RUN, INSTRUCTION IS LEGAL
118C  1E 0C       MVI  E,H'0C' } PRINT ERROR MESSAGE "ILLEGAL DIRECT"
118E  C3 97 04    JMP  X0497  }
-------------------------------------------------------------------------------------
11C1  CF    X11C1 RST  1      } SYNTAX CHECK FOR FN TOKEN
11C2  A8      DB 'A8' XRA  B  }
11C3  3E 80       MVI  A,H'80' LOAD ACC WITH 8Ø
11C5  32 01 03    STA  X0301 SET LOC Ø301 TO A NON-ZERO VALUE - FORCES INTO SYMBOL TABLE SEARCH AT ØF32-ØF38
11C8  B6          ORA  M  SET BIT 7 OF 1ST CHAR OF VARIABLE (NAME OF FUNCTION)
11C9  47          MOV  B,A PUT 1ST CHAR OF NAME IN REG B
11CA  C3 E5 0E    JMP  X0EE5 SEARCH THE SYMBOL TABLE TO FIND THE ADDR OF PTR TO THE FUNCTION DEFINITION
-------------------------------------------------------------------------------------
11CD  CD 91 21    CALL X2191 CONVERT ARG VALUE TO A CHAR STRING                    STR$
11D0  CD FC 11    CALL X11FC GENERATE CHAR COUNT BYTE FOR STRING BY COUNTING CHAR'S
11D3  CD 7A 13    CALL X137A REMOVE 3 BYTE IDENTITY GROUP FROM FORMULA TABLE, (CONVERTED # NOT IN STRING AREA - REST OF 137A IRRELEVANT)
11D6  01 CA 13    LXI  B,X13CA } PUT 13CA ON STACK FOR IMPLIED JMP AT 11ED  ADDR OF CHAR COUNT BYTE IN HL)
11D9  C5          PUSH B        }
11DA  7E    X11DA MOV  A,M PUT CHAR COUNT BYTE IN ACC
11DB  23          INX  H  ADVANCE PTR TO STRING FORMULA TABLE - OR IDENTITY GROUP AT Ø9ED
11DC  E5          PUSH H  SAVE ADDR OF PTR TO 1ST CHAR OF CONVERTED-ARG STRING
11DD  CD 58 12    CALL X1258 TEST IF ENOUGH ROOM IN STRING AREA FOR THE NEW STRING - PUT 3 BYTE IDENTITY GROUP IN STRING FORMULA TABLE
11E0  E1          POP  H  PUT ADDR OF PTR TO 1ST CHAR OF CONVERTED-ARG STRING IN HL  DE HAS ADDR IN STRING AREA FOR 1ST CHAR OF NEW STRING
11E1  4E          MOV  C,M } PUT ADDR OF 1ST CHAR OF CONVERTED-ARG STRING IN BC
11E2  23          INX  H   }
11E3  46          MOV  B,M }
11E4  CD F1 11    CALL X11F1 PUT 3-BYTE IDENTITY GROUP AT END OF FORMULA TABLE (IDENTITY PTR PTS TO STRING AREA, NOT WHERE STRING IS AT THIS MOMENT
11E7  E5          PUSH H  STORE ADDR Ø3C8 ON STACK
11E8  6F          MOV  L,A PUT CHAR COUNT BYTE IN L
11E9  CD 6D 13    CALL X136D MOVE CONVERTED-ARG STRING INTO STRING AREA
11EC  D1          POP  D  PUT ADDR Ø3C8 IN DE
11ED  C9          RET IMPLIED JMP TO 13CA - REMOVE IMPLIED JMP ADDR ØEØC (see ØE52-ØE55) FROM STACK, PUT PTR TO STRING TO BE EVALUATED IN HL
-------------------------------------------------------------------------------------
11EE  CD 58 12 X11EE CALL X1258 TEST IF ENOUGH ROOM IN THE STRING AREA FOR THE NEW STRING - PUT CHAR COUNT + PTR TO STRING SPACE IN
11F1  21 C8 03 X11F1 LXI  H,X03C8 LOAD HL WITH ADDR OF LAST 3 BYTE GROUP OF FORMULA TABLE  STRING FORMULA TABLE
11F4  E5          PUSH H PUT ADDR Ø3C8 ON STACK TO SPEED-UP INITIALIZING HL TO Ø3C8 AT 11FA
11F5  77          MOV  M,A STORE CHAR COUNT AT Ø3C8
11F6  23          INX  H  } STORE LO BYTE } PTR TO FIRST BYTE OF CHAR STRING
11F7  73          MOV  M,E }
11F8  23          INX  H  } STORE HI BYTE }
11F9  72          MOV  M,D }
11FA  E1          POP  H RESTORE HL TO POINT TO Ø3C8
11FB  C9          RET
-------------------------------------------------------------------------------------
11FC  2B    X11FC DCX  H BACKUP PTR TO LOOK AT 1ST CHAR
11FD  06 22 X11FD MVI  B,A'"' } PUT QUOTE CHAR IN B + D REG'S
11FF  50    X11FF MOV  D,B    } - TERMINATOR SEARCH CHAR IN B+D
1200  F5    X1200 PUSH H SAVE PTR TO CHAR STRING ON STACK - ADDR OF '"' IN FRONT OF STRING (OR 1 LESS THAN 1ST CHR)
1201  0E FF       MVI  C,H'FF' START CHAR COUNT AT -1
1203  23    >X1203 INX  H ADVANCE CHAR STRING PTR
1204  7E          MOV  A,M FETCH CHAR FROM STRING
1205  0C          INR  C INCREMENT CHAR COUNT
1206  B7          ORA  A TEST IF CHAR IS A NULL
1207  CA 12 12    JZ   X1212 JMP IF CHAR IS A NULL

                                                                                    2940
                                                                                    2941
                                                                                    2942
                                                                                    2943
                                                                                    2944
                                                                                    2945  TEST IF CURRENT LINE # IS
                                                                                    2946
                                                                                    2947  FFFF - IF IT IS, PROGRAM IS
                                                                                    2948  NOT RUNNING, AND CURRENT
                                                                                    2949
                                                                                    2950  INSTRUCTION IS ILLEGAL IN THE
                                                                                    2951  DIRECT MODE
                                                                                    2952
                                                                                    2953
                                                                                    2954
                                                                                    2955
                                                                                    2956
                                                                                    2957
                                                                                    2958
                                                                                    2959
                                                                                    2960
                                                                                    2961
                                                                                    2962
                                                                                    2963
                                                                                    2964
                                                                                    2965
                                                                                    2966
                                                                                    2967
                                                                                    2968
                                                                                    2969
                                                                                    2970
                                                                                    2971
                                                                                    2972
                                                                                    2973
                                                                                    2974
                                                                                    2975
                                                                                    2976
                                                                                    2977
                                                                                    2978
                                                                                    2979
                                                                                    2980
                                                                                    2981
                                                                                    2982
                                                                                    2983  PUT 3 BYTE GROUP IN STRING
                                                                                    2984  FORMULA TABLE - AT END OF
                                                                                    2985
                                                                                    2986  TABLE
                                                                                    2987
                                                                                    2988
                                                                                    2989
                                                                                    2990  COUNT CHAR'S IN STRING-VALUE
                                                                                    2991  IN STRING BEING EVALUATED
                                                                                    2992
                                                                                    2993  PUT CHAR COUNT + PTR TO STRING
                                                                                    2994  VALUE
                                                                                    2995
                                                                                    2996  IN STRING FORMULA TABLE
                                                                                    2997
                                                                                    2998
                                                                                    2999
```

```
                                ENTER AT 11FC OR 11FD
                          CMP   D     } JMP IF CHAR IS A `"'
                          JZ    X1212 }                          } OR look for a null if enter at
                          CMP   B     } JMP IF CHAR IS NOT `"'   } 11FF from 14E3
                          JNZ   X1203 }
X1212                     CPI   A'"'  } IF CHAR IS `"' CALL Ø7D1 - IF TERMINATOR CHAR TO BE FOUND IS A" NULL
                          CZ    X07D1 }
                          XTHL  PUT PTR TO END OF STRING ON STACK, PUT PTR TO START OF STRING IN HL (PTR TO `"' - SEE 12ØØ)
                          INX   H PT TO FIRST CHAR IN STRING
                          XCHG  PUT PTR TO 1ST CHAR OF STRING - VALUE IN DE
                          MOV   A,C PUT CHAR COUNT IN ACC
                          CALL  X11F1 STORE CHAR COUNT & PTR TO CHAR STRING IN FLAG AREA
X121E                     LXI   D,X03C8 LOAD DE WITH PTR TO FLAG AREA
PUSH D                    MVI   A,H'05' DUMMY
                          LHLD  X03A8 } LOAD HL WITH NEXT AVAILABLE BYTE IN STRING FORMULA TABLE
                          SHLD  X0412 } (ADDR OF CHAR COUNT BYTE OF STRING VALUE TO BE ADDED TO STRING FORMULA TABLE)
                          MVI   A,H'03' } SET # OF BYTES TO MOVE IN 1B66 TO Ø3
                          STA   X03A4 } SET VALUE TYPE CODE TO Ø3
                          CALL  X1B66 PUT 3 FLAGS IN TABLE Ø3AA - PUT 3 BYTE GROUP IN FORMULA TABLE
                          LXI   D,X03CB LOAD END-OF-TABLE ADDR INTO DE (1 MORE THAN END OF TABLE)
                          RST   4     TEST IF HL = DE
                          SHLD  X03A8 STORE ADDR OF NXT AVAILABLE POSITION IN TABLE Ø3AA
                          POP   H FETCH PTR TO END OF STRING FROM STACK (RESTORS HL TO PT TO STRING TO BE EVALUATED)
                          MOV   A,M PUT NULL IN ACC
                          RNZ   RETURN IF END OF TABLE NOT YET REACHED
                          MVI   E,H'10' } PRINT ERROR MESSAGE " STRING FORMULA TOO COMPLEX"
                          JMP   X0407 }
-------------------------------------------------------------------------------------------------------------------------------
X1240                     INX   H     ADVANCE PTR TO FIRST CHAR OF CONVERTED LINE #
X1241                     CALL  X11FC GENERATE CHAR COUNT BYTE FOR CHAR STRING SURROUNDED BY QUOTES AND/OR TERMINATED BY A NULL
X1244                     CALL  X137A REMOVE THE STRING-TO-PRINT FROM STRING AREA (IF IT'S THERE) - PUT PTR TO CHAR COUNT BYTE IN HL
                          CALL  X1B56 PUT 3 BYTE IDENTITY GROUP IN BCD - CHAR COUNT BYTE IN D
                          INR   D     ADVANCE CHAR CTR (TO ENTER LOOP)
X1248                     DCR   D     } DECREMENT CHAR COUNT - EXIT LOOP WHEN Ø
                          RZ          }
                          LOAX  B     FETCH CHAR FROM STRING
                          RST   3     PRINT CHAR ON OUTPUT CONSOLE
                          CPI   H'00' }
                          CZ    X0AA1 } PRINT NULLS AT END OF LINE (ONLY AFTER CR)
                          INX   B     ADVANCE PTR TO STRING TO PRINT
                          JMP   X1248 LOOP TIL ENTIRE STRING PRINTED
-------------------------------------------------------------------------------------------------------------------------------
X1258                     ORA   A     SET FLAGS - CHAR COUNT OF A STRING VALUE IN ACC
POP PSW                   MVI   C,H'F1' DUMMY
                          PUSH  PSH SAVE FLAGS & CHAR COUNT ON STACK
                          LHLD  X0300 LOAD HL WITH ADDR OF START OF STACK (1 BYTE PAST END OF STRING AREA)
                          XCHG        SAVE END-OF-STRING-AREA ADDR IN DE
                          LHLD  X03C9 LOAD HL WITH ADDR OF NEXT AVAILABLE BYTE IN STRING SPACE
                          CMA         } FORM 1'S COMPLEMENT OF CHAR COUNT - PUT IN BC
                          MOV   C,A   }
                          MVI   B,H'FF' } - # IS MINUS, THEREFORE B WILL BE FF
                          DAD   B     } COMPUTE NEW ADDR FOR NEXT BYTE AVAILABLE
                          INX   H     } - COMPLETES 2'S COMPLEMENT OF CHAR COUNT (HL HAS OLD ADDR - # NEW STRING CHAR COUNT)
                          RST   4     TEST ADDRESS JUST COMPUTED  RE HL < DE
                          JC    X1274 IF HL < DE, NEW ADDR < START OF STACK - NOT ENOUGH ROOM - PURGE OLD STRINGS FROM STRING AREA TO MAKE ROOM
                          SHLD  X03CB STORE NEW ADDR FOR NEXT AVAILABLE BYTE IN STRING AREA IN FLAG AREA
                          INX   H ADVANCE PTR TO STRING AREA - POINTS TO 1ST BYTE WHERE NEW STRING WILL BE PUT
                          XCHG  PUT PTR TO STRING AREA IN DE
X1272                     POP   PSH FETCH CHAR COUNT OF STRING FROM STACK
                          RET
-------------------------------------------------------------------------------------------------------------------------------
X1274                     POP   PSH PUT CHAR COUNT IN ACC & RESTORE FLAG STATUS FROM STACK
                          MVI   E,H'0E' }
                          JZ    X0407   } PRINT ERROR MESSAGE IF 2ND TIME THROUGH
```

```
:     3000
J     3001
8     3002
D     3003
      3004
      3005
      3006
#     3007
      3008
      3009
M     3010
 H    3011
>U    3012
*(    3013
      3014
>     3015
2$    3016
MTABLE 3017
      3018
      3019
"(    3020
      3021
      3022
@     3023
      3024
C7    3025
#     3026
M AND/3027
MTO CHA3028
MV    3029
      3030
      3031
H     3032
      3033
-     3034
      3035
L!    3036
      3037
CK    3038
7     3039
      3040
      3041
*}    3042
      3043
*K    3044
/     3045
0     3046
      3047
      3048
      3049
      3050
Z     3051
"K    3052
      3053
      3054
      3055
I     3056
      3057
      3058
J7    3059
```

(AND IT WAS FOUND AT 12ØE) - SCAN TIL NEXT NON-SPACE CHAR FOUND

SET-UP FOR ADDITION OF IDENTITY GROUP INTO STRING FORMULA TABLE

Ø3C8 Ø3C9 Ø3CA - LAST 3 LOC'S IN STRING FORMULA TABLE

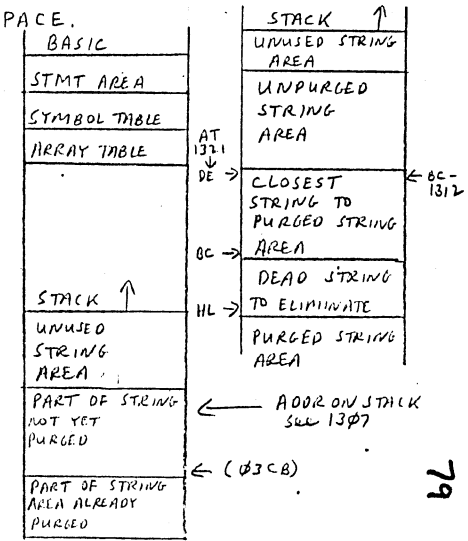
FREE-UP STRING SPACE BY ELIMINATING STRINGS WITHOUT AN IDENTITY GROUP IN STRING FORMULA TABLE, SYMBOL TABLE, OR ARRAY TABLE.

```
                                                    WHEN IMPLIED JMP TO 125A FROM 130E, 1274 WILL PRINT AN
                                                    ERROR MESSAGE
127A BF          CMP  A   TEST CHAR COUNT, CLEAR CARRY BIT - SET ZERO BIT (      ?     3060
1279 F5          PUSH PSW SAVE CHAR COUNT & FLAG STATUS ON STACK                       3061
127C 01 5A 12    LXI  B,X125A } PUT ADDR 125A ON STACK FOR IMPLIED JMP AT 130E  7     3062
127F C5          PUSH B       }                                                E     3063
1280 2A A6 03 X1280 LHLD X03A6 LOAD HL WITH ADDR OF 1ST BYTE OF STRING AREA     *&    3064
1283 22 C9 03 X1283 SHLD X03C9 STORE THIS VALUE IN PTR LOC FOR NEXT AVAILABLE BYTE IN UNPURGED "K  3065 STRING AREA
1286 21 00 00    LXI  H,X0000} LOAD HL WITH ORIGIN ADDR OF BASIC, PUT ORG ADDR ON STACK  !  — 3066 HOWEVER, THIS ADDR MUST BE 0000,
1289 E5          PUSH H      }                                                          3067 FLAG CHECK 130C - 130E IF STRING IDENTITY
128A 2A C0 03    LHLD X0300 } LOAD HL WITH ADDR OF START OF STACK (1 BYTE BELOW STRING AREA) *} 3068 GROUP NOT FOUND
128D E5          PUSH H     } PUT ADDR ON STACK                                        3069
128E 21 AA 03    LXI  H,X03AA LOAD HL WITH ADDR OF 1ST LOC IN STRING FORMULA TABLE  !*    3070
1291 EB       X1291 XCHG     SWAP PTRS                                                  3071
1292 2A A8 03    LHLD X03A8 LOAD HL WITH ADDR OF NEXT AVAILABLE BYTE IN STRING FORMULA TABLE *( 3072
1295 EB          XCHG       SWAP PTRS AGAIN                                          3073
1296 E7          RST  4     TEST PTRS AGAINST EACH OTHER                             3074
1297 01 91 12    LXI  B,X1291 LOAD BC WITH IMPLIED JMP ADDR - SIMILAR TO CALL 12EA AT 1287  3075
129A C2 E9 12    JNZ  X12E9 JMP IF STRING FORMULA TABLE IS NOT EMPTY (START AT 1ST GROUP IN TABLE B+ 'TEST 3076 ALL OTHER GROUPS - 1ST TO LAST)
-------------------------------------------------------------------------------------------------------------------------------------------
129D 2A E1 03    LHLD X03E1 LOAD HL WITH ADDR OF 1ST BYTE OF SYMBOL TABLE  *         3077
12A0 EB       X12A0 XCHG     SWAP PTRS                                                  3078
12A1 2A E3 03    LHLD X03E3 LOAD HL WITH ADDR OF 1ST BYTE OF ARRAY TABLE  *         3079
12A4 EB          XCHG       SWAP PTRS AGAIN, HL PTS TO 1ST BYTE OF SYMBOL TABLE, DE PTS TO 1ST BYTE 3080 OF ARRAY TABLE
12A5 E7          RST  4     TEST IF ENTIRE SYMBOL TABLE HAS BEEN SCANNED               3081
12A6 CA BE 12    JZ   X12BE JMP IF SYMBOL TABLE SCAN IS FINISHED  J>                  3082
12A9 7E          MOV  A,M   PUT VARIABLE SIZE BYTE OF VARIABLE IN ACC                  3083
12AA 23          INX  H   } ADVANCE PTR TO VARIABLE IN SYMBOL TABLE - PT TO 1ST BYTE OF  #  3084
12AB 23          INX  H   } VARIABLE VALUE REGION                                   #  3085
12AC 23          INX  H   }                                                         #  3086
12AD FE 03       CPI  H'03' TEST IF VARIABLE IS A STRING TYPE VARIABLE               3087
12AF C2 96 12    JNZ  X1296 JMP IF NOT A STRING VARIABLE                  B6        3088
12B2 CD EA 12    CALL X12EA TEST IDENTITY GROUP + SHUFFLE PTRS IF THIS IS THE NEXT NEAR-TO-TOP STRING  H  3089
12B5 AF          XRA  A     CLEAR ACC FOR DAD INSTR AT 12B9 (12EA ADVANCED PTR TO NXT VARIABLE IN TABLE) 3090
12B6 5F       X1286 MOV  E,A   PUT VARIABLE SIZE BYTE IN E                  -        3091
12B7 16 00       MVI  D,H'00' CLEAR D FOR DAD INSTR                                 3092
12B9 19          DAD  D     COMPUTE ADDR OF VARIABLE SIZE BYTE OF NEXT VARIABLE IN SYMBOL TABLE  3093
12BA C3 A0 12    JMP  X12A0 LOOP UNTIL ALL VARIABLES IN SYMBOL TABLE  C             3094
-------------------------------------------------------------------------------------------------------------------------------------------
12BD C1       X12BD POP  B   REMOVE PTR TO CURRENT ARRAY FROM STACK (see 12CC) - scan next array  A  3095
12BE EB       X12BE XCHG     SWAP PTRS                                               3096
12BF 2A E5 03    LHLD X03E5 LOAD HL WITH ADDR OF BYTE AFTER END OF ARRAY TABLE  *    3097
12C2 EB          XCHG SWAP PTRS AGAIN, HL PTS TO 1ST BYTE OF ARRAY TABLE, DE PTS TO BYTE AFTER END OF ARRAY TABLE 3098
12C3 E7          RST  4 TEST IF ENTIRE ARRAY TABLE HAS BEEN SCANNED                 3099
12C4 CA 0A 13    JZ   X130A JMP IF ARRAY TABLE SCAN IS FINISHED  J                  3100
12C7 7E          MOV  A,M   PUT ARRAY VARIABLE - SIZE BYTE IN AC                    3101
12C8 23          INX  H     ADVANCE PTR TO ARRAY TABLE  #                           3102
12C9 CD 54 18    CALL X1854 PUT 4 BYTES IN BCDE, 2 BYTES IN BC CONTAIN # OF BYTES IN ARRAY MT ( 3103 see pg 71)
12CC E5          PUSH H     PUT PTR TO CURRENT ARRAY ON STACK                       3104
12CD 09          DAD  B     COMPUTE ADDR OF VARIABLE SIZE BYTE OF NEXT ARRAY IN ARRAY TABLE  3105
12CE FE 03       CPI  H'03' TEST IF ARRAY IS A STRING TYPE ARRAY                    3106
12D0 C2 BD 12    JNZ  X12BD JMP IF NOT A STRING ARRAY  B=                           3107
12D3 22 CD 03    SHLD X03CD SAVE ADDR FOR NEXT ARRAY IN PTR LOC IN FLAG AREA  "M    3108
12D6 E1          POP  H     PUT PTR TO CURRENT ARRAY IN HL                          3109
12D7 4E          MOV  C,M   PUT # OF DIMENSIONS BYTE IN C  N                        3110
12D8 06 00       MVI  B,H'00' CLEAR B FOR DAD INSTRUCTIONS                          3111
12DA 09          DAD  B   } COMPUTE ADDR OF 1ST BYTE OF 1ST ARRAY ELEMENT VARIABLE VALUE REGION  3112
12DB 09          DAD  B   } (skip over arg-element ctr's - see page 71)             3113
12DC 23          INX  H   }                                                         #  3114
12DD EB       X12DD XCHG     SWAP PTRS                                               3115
12DF 2A CD 03    LHLD X03CD LOAD HL WITH ADDR OF VARIABLE-SIZE BYTE OF NEXT ARRAY  *M  3116
12E1 EB          XCHG SWAP PTRS AGAIN, HL PTS TO ELEMENTS IN CURRENT ARRAY, DE HAS ADDR 1 BYTE BEYOND END 3117 OF CURRENT ARRAY
12E2 E7          RST  4 TEST PTRS AGAINST EACH OTHER                                3118
12E3 CA 4E 12    JZ   X12DE JMP IF ALL ELEMENTS OF CURRENT ARRAY HAVE BEEN 'TESTED' BY 12EA  J>  3119
```

SYMBOL TABLE

VARIABLE SIZE
2ND CHAR } 2 BYTE NAME
1ST CHAR }
VARIABLE VALUE REGION

```
12E6  01 00 12          LXI  B,X1200 FOR ARRAYS, PUT IMPLIED JMP ADDR ON STACK (SIMILAR TO CALL )IDEA 3120 AT 12B2)
12E9  C5          X12E9 PUSH B  PUT IMPLIED JMP ADDR ON STACK                          E     3121
12EA  AF          X12EA XRA  A CLEAR ACC                                               /     3122
12EB  96                ORA  M PUT CHAR COUNT BYTE FROM IDENTITY GROUP (IN VARIABLE REGION) IN ACC 6 - SET 3123 FLAGS - TESTS FOR NULL STRING
12EC  23                INX  H  } PUT PTR TO 1ST CHAR OF STRING IN DE                #     3124
12ED  5E                MOV  E,M }                                                   ^     3125
12EE  23                INX  H                                                       #     3126
12EF  56                MOV  D,M                                                     V     3127
12F0  23          1291, 1285 INX  H  ADVANCE PTR TO POINT TO 1ST BYTE OF NEXT IDENTITY GROUP  #  3128
12F1  C8          or 1200  RZ   IF THE STRING IS A NULL STRING, NO NEED TO ELIMINATE STRING - RETURN  H  3129
12F2  44                MOV  B,H } SAVE PTR TO VARIABLE REGION IN BC - ADDR OF NEXT IDENTITY GROUP  C  3130
12F3  4D                MOV  C,L }                  IN MEMORY (FORMULA, SYMBOL, 1ST ARRAY  M  3131 TABLES)
12F4  2A C8 03          LHLD X03C8 LOAD HL WITH ADDR OF NEXT AVAILABLE BYTE IN STRING AREA (REMEMBER  *K  3132 1288-1283) see picture below
12F7  E7                RST  4  TEST IF STRING (IDENTITY GROUP IN VARIABLE) IS IN STRING SPACE  UNPURGED  3133 STRING AREA
12F8  60                MOV  H,B } PUT PTR TO VARIABLE REGION IN HL                          3134
12F9  69                MOV  L,C }                                                           3135
12FA  D8                RC   RETURN IF STRING IS IN PART OF STRING AREA ALREADY PURGED  HL < DE  X  3136
12FB  E1                POP  H PUT RETURN ADDR IN HL                                         3137
12FC  E3                XTHL PUT ADDR OF END OF STRING AREA IN HL (see 128D) PUT RETURN ADDR ON BACK ON STACK  3138
12FD  E7                RST  4          or ADDR OF 1ST CHAR OF STRING CLOSEST TO PURGED STRING  AREA  3139
12FE  E3                XTHL } PUT ADDR OF END OF STRING AREA BACK ON STACK ABOVE RETURN ADDR  3140
12FF  E5                PUSH H }          ADDR OF 1ST CHAR OF STRING CLOSEST TO PURGED STRING AREA  3141
1300  60                MOV  H,B } PUT PTR TO VARIABLE REGION IN HL - ADDR OF NEXT IDENTITY GROUP IN MEMORY  3142 see 12F2
1301  69                MOV  L,C }                                                           3143
1302  D0                RNC  RETURN IF STRING IS IN THE STMT AREA (MEMORY LOWER THAN START OF PSTACK)  3144
1303  C1                POP  B  PUT RETURN ADDR IN BC                                A       3145  STRING IS IN STRING SPACE, BUT (03C8)
1304  F1                POP  PSW REMOVE ADDR OF END OF STRING AREA FROM STACK (see 128D) } INFO FOR STRING CLOSEST TO PURGED  3146  HAS NOT BEEN UPDATED TO INCLUDE IT
1305  F1                POP  PSW REMOVE ADDR OF ORIGIN OF BASIC FROM STACK (see 1289) }  STRING AREA  3147
1306  E5                PUSH H  PUT ADDR OF VARIABLE VALUE REGION ON STACK (or POSITION IN STRING FORMULA TABLE)  3148
1307  D5                PUSH D  PUT ADDR OF 1ST CHAR OF STRING ON STACK - THIS STRING IS CLOSEST TO PURGED STRING AREA  D  3149
1308  C5                PUSH B  PUT RETURN ADDR BACK ON STACK                        E       3150
1309  C9                RET                                                          I       3151
130A  D1          X130A POP  D  PUT ADDR OF 1ST CHAR OF STRING IN DE - THIS STRING IS CLOSEST TO PURGED STRING AREA  3152 STARTING AT BEGINNING OF STRING
130B  E1                POP  H  PUT ADDR OF IDENTITY GROUP IN HL                             3153 AREA, A STRING NEAREST TOP OF
130C  7D                MOV  A,L } TEST IF ADDR OF IDENTITY GROUP IS 0000                    3154
130D  B4                ORA  H } LAST STRING HAS BEEN MOVED - DONE WITH PURGE        4       3155 STRING AREA HAS BEEN FOUND. IF
130E  C8          TO 125A RZ   JMP TO 125A - see 127C - 127E                         H       3156 IT HAS AN IDENTITY GROUP, IT MUST
130F  2B                DCX  H  BACK-UP PTR TO IDENTITY GROUP (MATCHES INX H AT 12F0)  +       3157 CONTINUE TO EXIST. IF NOT,
1310  46                MOV  B,M } PUT ADDR OF 1ST CHAR OF STRING IN BC              F       3158 ELIMINATE IT TO FREE UP STRING
1311  2B                DCX  H  }                                                    +       3159 SPACE.
1312  4E                MOV  C,M }                                                   N       3160
1313  E5                PUSH H  SAVE PTR TO IDENTITY GROUP ON STACK (ADDR OF PTR ADDR)       3161
1314  2B                DCX  H  } PUT CHAR COUNT BYTE OF STRING IN L                 +       3162
1315  6E                MOV  L,M }                                                           3163
1316  26 00             MVI  H,H'00' CLEAR H FOR DAD INSTRUCTION                     &       3164
1318  09                DAD  B  COMPUTE ADDR OF BYTE AFTER LAST BYTE OF STRING               3165
1319  50                MOV  D,B } PUT ADDR OF 1ST BYTE OF STRING IN DE              P       3166
131A  59                MOV  E,C }                                                   Y       3167
131B  2B                DCX  H  } PUT ADDR OF LAST BYTE OF STRING IN BC              +       3168
131C  44                MOV  B,H } REMEMBER ADDING CHAR COUNT TO 1ST BYTE ADDR AT 1318  D    3169
131D  4D                MOV  C,L }                                                   M       3170
131E  2A C8 03          LHLD X03C8 LOAD HL WITH ADDR OF CURRENT POSITION IN STRING AREA  *K  3171
1321  CD 7B 04          CALL X047B MOVE CLOSEST STRING UP TO PURGED STRING AREA      M       3172
1324  E1                POP  H  PUT ADDR WHERE PTR TO CHAR STRING IS STORED IN HL (ADDR OF IDENTITY GROUP)  3173
1325  71                MOV  M,C } PUT NEW ADDR OF 1ST CHAR OF STRING JUST MOVED INTO IDENTITY GROUP IN  3174
1326  23                INX  H  } MEMORY                                             #       3175
1327  70                MOV  M,B }                                                           3176
1328  69                MOV  L,C } PUT ADDR 1 BYTE LOWER IN HL - THIS IS THE NEW ADDR FOR NEXT BYTE OF  3177
1329  60                MOV  H,B } UNPURGED STRING AREA                                      3178
132A  2B                DCX  H  }                                                    +       3179
```



```
1328  C3 83 12          JMP   X1283 LOOP UNTIL ENTIRE STRING AREA HAS BEEN PURGED OF 'DEAD' C   3180 STRINGS
132B  C5          X132E   PUSH  B STORE PRECEDENCE BYTE OF PREVIOUS OPERATOR ON STACK (see ØC64) E   3181 STRING VALUE CONCATENATION
132F  E5                  PUSH  H  } PUT PTR TO CHAR COUNT OF PREVIOUSLY EVALUATED STRING VALUE     3182   (ØC98)
1330  2A 12 04            LHLD  X0412 } ON STACK (see 1226) - call this value 1ST ARG *   3183
1333  E3                  XTHL      } LEAVE PTR TO STRING TO BE EVALUATED IN HL                    3184
1334  CD C8 00            CALL  X00C8 EVALUATE NEXT PART OF STRING (PART OF STRING AFTER '+') - PUT 3 MK MORE 3185 BYTES IN STRING FORMULA TABLE
1337  E3                  XTHL  PUT ADDR OF CHAR COUNT BYTE OF 1ST ARG IN HL, PUT PTR TO STRING TO BE EVALUATED ON 3186   └VALUE FOUND CALLED 2ND ARG
1338  CC AB 1C            CALL  X1CAB TEST THAT 2ND ARG IS A STRING VALUE  M STACK 3187
133B  7E                  MOV   A,M  PUT CHAR COUNT OF 2ND ARG IN ACC              3188
133C  E5                  PUSH  H    SAVE ADDR OF CHAR COUNT BYTE OF 1ST ARG ON STACK 3189
133D  2A 12 04            LHLD  X0412 LOAD HL WITH ADDR OF CHAR COUNT OF 2ND ARG (see 1226) * 3190
1340  E5                  PUSH  H    SAVE ADDR OF CHAR COUNT BYTE OF 2ND ARG ON STACK 3191
1341  8E                  ADD   M    ADD CHAR COUNT OF 2ND ARG TO ACC              3192
1342  1E 0F               MVI   E,H'0F' } IF COMBINED CHAR COUNT PRODUCES AN OVERFLOW 3193
1344  DA 97 04            JC    X0497 } PRINT ERROR MESSAGE "STRING TOO LONG"  Z7  3194
1347  CD EE 11            CALL  X11EE TEST IF ENOUGH ROOM IN STRING AREA FOR CONCATENATED STRING, PUT CHAR M COUNT 3195 + PTR BYTES IN Ø3C8 Ø3C9 Ø3CA
134A  D1                  POP   D    PUT ADDR OF CHAR COUNT BYTE OF 2ND ARG IN DE (DE HAS ADDR-PTS TO FO M MULA 3196 TABLE)
134B  CD 7E 13 (MAYBE NECESSARY) CALL X137E REMOVE 3 BYTE GROUP FROM FORMULA TABLE, CHANGE NXT-BYTE PTR FOR STRING M AREA 3197 IF 2ND ARG LAST STRING ENTERED IN AREA
134E  E3 TYPE OF CALL     XTHL  PUT ADDR OF CHAR COUNT BYTE OF 1ST ARG IN HL, PUT ADDR OF CHAR COUNT BYTE OF 2ND ARG 3198 ON STACK
134F  CD 70 13            CALL  X1370 REMOVE 3 BYTE GROUP FROM FORMULA TABLE, CHANGE NXT-BYTE PTR FOR STRING MAREA 3199 IF 1ST ARG LAST STRING IN STRING AREA AT
1352  E5                  PUSH  H PUT ADDR OF CHAR COUNT BYTE OF 1ST ARG ON STACK  3200   THE MOMENT (see 1391)
1353  2A C9 03            LHLD  X03C9 } PUT ADDR WHERE CONCATENATED STRING IS TO BE PUT IN DE (ADDR STORED * IN 3201
1356  EB                  XCHG        } LAST LOC'S OF FORMULA TABLE IN 11EE AT 1347 (see 11F1 - 11FA)) 3202
1357  CD 65 13            CALL  X1365 MOVE 1ST ARG STRING VALUE INTO STRING AREA  M  3203
135A  CD 65 13            CALL  X1365 MOVE 2ND ARG STRING VALUE INTO STRING AREA   M  3204
135D  21 64 0C            LXI   H,X0C64 } PUT ADDR ØC64 ON STACK ABOVE PTR TO  SP → ! 3205
1360  E3                  XTHL          } STRING BEING EVALUATED (SEE 1200 or 1222)    3206
1361  E5                  PUSH  H       } THIS ADDR IS IMPLIED JMP BACK INTO STRING     3207
1362  C3 1E 12            JMP   X121E PUT 3 BYTE GROUP FOR CONCATENATED STRING IN FORMULA TABLE C 3208
                                  EVALUATOR AT 123A
```

STACK BEFORE 135D - 1361: PTR TO STRING

STACK AFTER 135D - 1361: ØC64; PTR TO STRING ← SP

```
1365  E1          X1365   POP   H  PUT RETURN ADDR IN HL                          3209  MOVE A CHAR STRING INTO STRING
1366  E3                  XTHL  PUT RETURN ADDR BACK ON STACK AFTER REMOVING PTR TO STRING FORMULA TABLE IN 3210 HL
1367  7E                  MOV   A,M FETCH A CHAR COUNT BYTE FROM THE TABLE         3211  AREA, DE HAS DEST ADDR PRIOR,
1368  23                  INX   H  } FETCH AN ADDR (POINTS TO 1ST BYTE OF A SELECTED CHAR STRING) # 3212  REMOVES 1 ADDR FROM STACK
1369  4E                  MOV   C,M } FROM TABLE INTO BC                         N  3213
136A  23                  INX   H                                                #  3214  ADDR ON STACK SHOULD POINT
136B  46                  MOV   B,M                                              F  3215
136C  6F                  MOV   L,A PUT CHAR COUNT IN L                             3216  TO ADDR CONTAINING A CHAR COUNT
136D  2C            X136D INR   L  INCR CHAR COUNT TO ENTER LOOP                  ,  3217  FOLLOWED BY A 2 BYTE PTR ADDR
136E  2D            X136E DCR   L  DECR CHAR COUNT                               -  3218
136F  C8                  RZ    ALL CHAR'S MOVED WHEN CHAR CTR IS Ø - DONE       H  3219  (PTS TO CHAR STRING)
1370  0A                  LDAX  B } MOVE 1 BYTE, SOURCE ADDR IN BC, DESTINATION ADDR IN DE 3220
1371  12                  STAX  D }                                                3221
1372  03                  INX   B ADVANCE SOURCE PTR                               3222
1373  13                  INX   D ADVANCE DESTINATION PTR                          3223
1374  C3 6E 13            JMP   X136E LOOP UNTIL ALL CHAR'S HAVE BEEN MOVED        C  3224
1377  CD 98 1C    X1377   CALL  X1C88 PERFORM STRING VALUE CHECK ON VALUE IN HOLDING AREA - PRINT ERROM 3225 MESSAGE "TYPE MISMATCH" IF NOT STRING TYPE
137A  2A 12 04    X137A   LHLD  X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF LAST GROUP PUT IN FORMULA * TABLE 3226 REMOVE A STRING FROM STRING AREA
137D  EB          X137D   XCHG        PUT PTR TO CHAR COUNT BYTE OF LAST GROUP IN DE  FORCES REMOVAL OF GROUP FROM TABLE 3227 BY CHANGING NXT-BYTE PTR
137E  CD 96 13    X137E   CALL  X1396 REMOVE A 3 BYTE GROUP FROM STRING FORMULA TABLE (FETCH GROUP, M BUT 3228 NOT REMOVE, IF DE ≠ ADDR OF CHAR COUNT
1381  EB                  XCHG  PUT ADDR OF CHAR COUNT BYTE OF GROUP REMOVED FROM TABLE IN DE 3229 BYTE OF GROUP IN TABLE
1382  C0                  RNZ   EXIT HERE IF THE GROUP WAS NOT REMOVED FROM THE FORMULA TABLE (see 139F) @ 3230
1383  D5                  PUSH  D PUT ADDR OF CHAR COUNT BYTE OF REMOVED GROUP ON STACK U 3231
1384  50                  MOV   D,B } PUT PTR ADDR OF GROUP REMOVED FROM TABLE IN DE P 3232
1385  59                  MOV   E,C } (POINTS TO 1ST BYTE IN THE RELATED STRING)     Y 3233
1386  1B                  DCX   D BACK-UP PTR TO MAKE IT NXT-BYTE AVAILABLE TYPE OF ADDR (USED IN COMPARISON TO 3234 DETERMINE IF STRING LATEST ONE IN STRING AREA)
1387  4E                  MOV   C,M PUT CHAR COUNT OF REMOVED GROUP IN C - USED BY DAD IN 1398) N 3235 REMEMBER STRINGS ARE UPSIDE DOWN 1ST CHAR
1388  2A CB 03            LHLD  X03CB LOAD HL WITH NXT-BYTE AVAILABLE IN STRING SPACE ADDR * K 3236 IS IN LOWER MEMORY THAN LAST CHAR
138B  E7                  RST   4    TEST IF PTR ADDR'S MATCH                      3237
138C  C2 94 13            JNZ   X1394 IF PTR' ADDR'S MATCH, GROUP REMOVED SAME AS LAST GROUP PUT IN B TABLE 3238
138F  47                  MOV   B,A CLEAR B REG FOR DAD                         G  3239
```

```
1390 09          DAD  B COMPUTE NEW NXT-BYTE AVAILABLE ADDR                    3240
1391 22 CB 03    SHLD X03CB STORE THIS NEW ADDR IN FLAG AREA (THIS EFFECTIVELY FREES SPACE IN STRING AREA)  3241
1394 E1    X1394 POP  H PUT ADDR OF CHAR COUNT BYTE OF REMOVED GROUP IN HL     3242
1395 C9          RET                                                          I  3243
-------------------------------------------------------------------------------------
1396 2A A8 03 X1396 LHLD X03A8 FETCH PTR TO CURRENT POSITION IN STRING FORMULA TABLE FROM * (FLAG 3244 AREA
1399 2B          DCX  H    BACK UP PTR                                        +  3245  REMOVE A 3 BYTE GROUP FROM
139A 46          MOV  B,M  } FETCH 2 BYTES FROM TABLE INTO, BC REG'S          F  3246
139B 2B          DCX  H                                                       +  3247  STRING FORMULA TABLE
139C 4E          MOV  C,M  } (PUT AN ADDR IN BC, HL POINTS TO CHAR COUNT IN   N  3248
139D 2B          DCX  H    }       TABLE AFTER 139D)                          +  3249  DE CONTAINS AN ADDR, WHICH, IF
139E E7          RST  4 TEST IF HL = DE                                          3250  SAME AS ADDR OF CHAR COUNT BYTE
139F C0          RNZ  IF HL ≠ DE RETURN                                       @  3251  OF GROUP LAST PUT IN TABLE, EXITS
13A0 22 A8 03    SHLD X03A8 ABOVE OPERATION REMOVED 3 BYTES FROM THE TABLE, STORE NEW ADDR IN 3252  AT 13A3 (see 134A)
13A3 C9          RET  TABLE PTR (DE HAD TO CONTAIN ADDR OF CHAR COUNT BYTE OF THIS GROUP)  I  3253
-------------------------------------------------------------------------------------
13A4 01 04 10    LXI  B,X1004 } PUT ADDR 1004 ON STACK FOR IMPLIED JMP AT 13AF   T  3254  LEN
13A7 C5          PUSH B       }                                               E  3255
                                        LEN
13A8 CD 77 13 X13A8 CALL X1377 SAME AS FOR LEN ──→ IF ARG STRING LAST IN AREA, REMOVE IT, ADDR M OF 3256 CHAR COUNT BYTE IS IN HL
13AB AF          XRA  A  } CLEAR D (FOR DAD AT 1498)  CLEAR ACC              /  3257
13AC 57          MOV  D,A}                                                   W  3258
13AD 7E          MOV  A,M FETCH LENGTH BYTE FROM STRING FCN TABLE  FETCH LENGTH BYTE FROM STRING FORMULA TABLE  3259
13AE B7          ORA  A SET FLAGS - TEST FOR NULL STRING                        3260
13AF C9          RET  (used by 1399-13B3)  IMPLIED JMP TO ROUTINE TO CONVERT BYTE IN ACC TO 3261 2 BYTE INTEGER IN HOLDING AREA
-------------------------------------------------------------------------------------
13B0 CD AA 13    CALL X13AA IF ARG STRING LAST STRING IN STRING AREA, REMOVE IT, ADDR OF CHAR COUNT BYTE IN M ( HL  3262  ASC
13B3 CA EE 08    JZ   X08EE ZERO FLAG SET-IN 13AF IF NULL STRING - PRINT ERROR MESSAGE "ILLEGAL FUNCTION  3263 CALL
13B6 23          INX  H ADVANCE PTR TO STRING FCN TABLE - POINT TO STRING PTR ADDR IN TABLE  #  3264
13B7 5E          MOV  E,M } FETCH PTR TO STRING INTO DE FROM STRING FCN TABLE  ^  3265
13B8 23          INX  H   }                                                  #  3266
13B9 56          MOV  D,M }                                                  V  3267
13BA 1A          LDAX D FETCH 1ST CHAR OF STRING INTO ACC                       3268
13BB C3 04 10    JMP  X1004 CONVERT CHAR TO 2 BYTE INTEGER VALUE IN HOLDING AREA  CT  3269
-------------------------------------------------------------------------------------
13BE 3E 01       MVI  A,H'01' PUT A CHAR COUNT OF 1 IN ACC (FCN PRODUCES A 1 CHAR STRING)  >  3270  CHR$
13C0 CD EE 11    CALL X11EE TEST IF ENOUGH ROOM IN STRING AREA FOR 1 CHAR STRING, PUT IDENTITY GROUP AT END M OF 3271 FORMULA TABLE
13C3 CD 84 14    CALL X1484 CONVERT ARG IN HOLDING AREA TO A 1 BYTE INTEGER (0-255)  M  3272
13C6 2A C9 03    LHLD X03C9 PUT ADDR WHERE STRING STARTS IN HL (PUT THERE BY 11F1-11F9)  *I  3273
13C9 73          MOV  M,E  STORE THE 1-BYTE ARG IN THE STRING AREA              3274
13CA C1    X13CA POP  B REMOVE IMPLIED JMP ADDR 0E0C FROM STACK (see 0E52-0E55) - NOT NEEDED A FOR 3275 THIS FCN
13CB C3 1E 12    JMP  X121E MOVE IDENTITY GROUP INTO STRING FORMULA TABLE, PUT PTR TO STRING TO C BE  3276 EVALUATED IN HL
-------------------------------------------------------------------------------------
13CE CD 5A 14    CALL X145A PERFORM SYNTAX CHECK FOR ')', MOVE 2ND ARG FROM STACK TO B  MZ  3277  LEFT $
1301 AF          XRA  A    CLEAR ACC (FLAG FOR LEFT$ FCN) - CHAR'S START AT THE LEFT, OFFSET OF 0 3278 (see 13EE)
1302 E3    X1302 XTHL PUT PTR TO CHAR COUNT BYTE IN HL, PUT PTR TO STRING TO BE EVALUATED ON STACK  3279
1303 4F          MOV  C,A SAVE OFFSET FOR 1ST CHAR POSITION IN STRING INC        0  3280
1304 3E E5       PUSH H MVI A,H'E5' Dummy                                       >  3281
1306 E5    X1306 PUSH H SAVE PTR TO CHAR COUNT BYTE OF 1ST ARG (STRING) ON STACK  3282
1307 7E          MOV  A,M PUT CHAR COUNT BYTE IN ACC (FROM STRING FORMULA TABLE)  3283
1308 B8          CMP  B COMPARE CHAR COUNT AGAINST 2ND ARG                      8  3284
1309 DA 0E 13    JC   X130E JMP IF 2ND ARG > CHAR COUNT OF 1ST ARG              Z^  3285
130C 78          MOV  A,B IF 2ND ARG ≤ CHAR COUNT, THEN STRING FROM FCN HAS CHAR COUNT = 2ND ARG  3286 PUT THIS NEW CHAR COUNT IN ACC FOR CALL 1258
130D 11 0E 00 MVI C,H'00' LXI D,X000E Dummy                                     3287
1310 C5          PUSH B SAVE BC ON STACK (2ND ARG + BRANCH FLAG)                E  3288
1311 CD 58 12    CALL X1258 TEST IF ENOUGH ROOM IN STRING AREA FOR THE NEW STRING, ADDR OF 1ST CHAR OF 3289 NEW STRING (IN STRING AREA) IN DE
1314 C1          POP  B RESTORE BC FROM STACK - B IS 2ND ARG C IS OFFSET FOR 1ST CHAR POSITION IN 3290 STRING
1315 E1          POP  H } PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN HL, LEAVE IT ON THE STACK  3291
1316 E5          PUSH H }                                                       3292
1317 23          INX  H ADVANCE PTR TO STRING FORMULA TABLE                     #  3293
1318 46          MOV  B,M } PUT PTR TO 1ST CHAR OF 1ST ARG STRING IN HL         F  3294
1319 23          INX  H   }                                                     #  3295
131A 66          MOV  H,M }                                                     3296
131B 69          MOV  L,B }                                                     3297
131C 06 00       MVI  B,H'00' CLEAR B FOR DAD INSTR                             3298
131E 09          DAD  B COMPUTE ADDR OF 1ST CHAR OF STRING AS A RESULT OF THE FCN BEING 3299
                                                                  EXECUTED
```

```
13EF  44           MOV  B,H  } PUT ADDR OF 1ST CHAR OF NEW STRING (RESULT OF FCN) IN BC   D  3300
13F0  40           MOV  C,L  }                                                           M  3301
13F1  CD F1 11     CALL X11F1 PUT 3-BYTE IDENTITY GROUP IN LAST POSITION OF STRING FORMULA TABLE  3302
13F4  6F           MOV  L,A  PUT CHAR COUNT OF NEW STRING IN L                           3303
13F5  CD 6D 13     CALL X136D MOVE NEW STRING INTO STRING AREA                        M  3304
13F8  D1           POP  D    PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN DE             Q  3305
13F9  CD 7E 13     CALL X137E REMOVE THE IDENTITY GROUP FROM FORMULA TABLE IF LAST ONE ENTERED  M  3306
13FC  C3 1E 12     JMP  X121E PUT 3-BYTE IDENTITY GROUP FOR NEW STRING IN STRING FORMULA TABLE  C  3307

                                                                                                  3308   RIGHT$
13FF  CD 5A 14     CALL X145A PERFORM SYNTAX CHECK FOR ')', MOVE 2ND ARG FROM STACK TO BC  MZ  3308
1402  D1           POP  D  } PUT PTR TO CHAR COUNT BYTE IN DE ; LEAVE IT ON STACK        Q  3309
1403  D5           PUSH D  }   (OF 1ST ARG)                                              U  3310
1404  1A           LDAX D  PUT CHAR COUNT OF 1ST ARG (STRING) IN ACC                        3311
1405  90           SUB  B  SUBTRACT 2ND ARG - COMPUTE OFFSET FOR 1ST CHAR POSITION OF NEW STRING  3312
1406  C3 02 13     JMP  X1302 REST OF THIS FCN SIMILAR TO LEFT$                          CR  3313

                                                                                                  3314   MID$
1409  EB           XCHG PUT PTR TO STRING TO BE EVALUATED IN HL (WAS PUT IN DE AT ØE3C)      3315
140A  7E           MOV  A,M PUT DELIMITER CHAR IN STRING TO BE EVALUATED IN ACC (CHAR IS ')' IF ONLY 2 ARGS; CHAR IS ',' IF 3 ARGS)
140B  CD 50 14     CALL X1450 PUT 2ND ARG IN B, TEST THAT 2ND ARG IS NOT Ø                M  3316
140E  C5           PUSH B    PUT 2ND ARG ON STACK                                        E  3317
140F  1E FF        MVI  E,H'FF' SET 3RD ARG TO MAXIMUM IF 2ND ARG NOT IN STRING BEING EVALUATED  3318
1411  FE 29        CPI  A')' TEST DELIMITER CHAR                                          )  3319
1413  CA 1B 14     JZ   X141B JMP IF CHAR IS ')' - ONLY 2 ARGS INVOLVED                   J  3320
1416  CF           RST  1  } PERFORM SYNTAX CHECK FOR ',' BETWEEN 2ND & 3RD ARGS          O  3321
1417  2C           ~~INR L~~ }                                                            ,  3322
1418  CD 81 14     CALL X1481 EVALUATE 3RD ARG TO A 1 BYTE INTEGER VALUE IN E             M  3323
141B  CF           RST  1  } PERFORM SYNTAX CHECK FOR ')'                                 O  3324
141C  29           ~~DAD H~~ }                                                            )  3325
141D  F1           POP  PSW PUT 2ND ARG IN ACC                                              3326
141E  E3           XTHL PUT PTR TO CHAR COUNT BYTE OF 1ST ARG (STRING) IN HL (see ØE33-ØE37), PUT PTR TO STRING TO BE EVALUATED ON STACK  3327
141F  01 06 13     LXI  B,X1306 } PUT ADDR 1306 ON STACK FOR IMPLIED JMP AT 1427, 1420, OR 142F  V  3328
1422  C5           PUSH B        }                                                        E  3329
1423  3D           DCR  A  DECR 2ND ARG - 1ST CHAR POSITION OFFSET                        =  3330
1424  BE           CMP  M  COMPARE 2ND ARG AGAINST BYTE COUNT OF 1ST ARG STRING           >  3331
1425  06 00        MVI  B,H'00' FORCE 2ND ARG TO Ø (JUST IN CASE 2ND ARG > CHAR COUNT OF 1ST ARG)  3332
1427  D0   1306 <- RNC  IMPLIED JMP TO 1306 IF 2ND ARG > CHAR COUNT OF 1ST ARG          P  3333
1428  4F           MOV  C,A  PUT 2ND ARG-1 IN C                                           O  3334
1429  7E           MOV  A,M  PUT CHAR COUNT BYTE IN ACC                                     3335
142A  91           SUB  C  COMPUTE # OF CHAR'S IN ARG STRING STARTING AT 2ND ARG POSITION   3336
142B  B9           CMP  E  COMPARE THIS REMAINDER WITH 3RD ARG                            :  3337
142C  47           MOV  B,A } USE ENTIRE ARG STRING STARTING AT 2ND ARG CHAR POSITION     G  3338
142D  D8   1306 <- RC       } SINCE 3RD ARG > REMAINDER CHAR COUNT OF ARG STRING        X  3339
142E  43           MOV  B,E } USE ONLY 3RD-ARG # OF CHAR'S, SINCE 3RD ARG ≤ REMAINDER CHAR  3340
142F  C9   1306 <- RET      }   COUNT OF ARG STRING                                    C  3341

                                                                                          I  3342   INP
1430  CD A4 14     CALL X14A4 CONVERT LVALUE IN HOLDING TO 1 BYTE INTEGER IN ACC          M  3343
1433  32 37 14     STA  X1437 STORE ARGUMENT IN INPUT INSTRUCTION AS PORT #              27  3344
1436  DB 00        IN   H'00' FETCH A BYTE FROM THE INPUT PORT SPECIFIED BY THE ARGUMENT  3345   AREA
1438  C3 04 10     JMP  X1004 CONVERT BYTE IN ACC TO INTEGER FORMAT IN HL - STORE IN HOLDING  3346

                                                                                          M  3346   OUT
143B  CD 74 14     CALL X1474 CONVERT 1ST ARG TO PORT #, STORE IN NEXT INSTRUCTION, 2ND ARG  3347
143E  D3 00        OUT  H'00' CONVERTED TO 1 BYTE IN ACC.                                 I  3348
1440  C9           RET        -> OUTPUT BYTE TO PORT                                      B IN ACC  3349

                                                                                                  3350   WAIT
1441  CD 74 14     CALL X1474 CONVERT 1ST ARG TO PORT #, STORE AT 1453, 2ND ARG CONVERT TO 1 BYTE  3351
1444  F5           PUSH PSW  2ND ARGUMENT IS STATUS MASK, SAVE ON STACK                   +  3352
1445  1E 00        MVI  E,H'00' IF NO 3RD ARGUMENT IN INSTRUCTION, DEFAULT 3RD ARG TO ØØ  W  3353
1447  2B           DCX  H  BACK-UP PTR TO INSTRUCTION                                     A  3354
1448  D7           RST  2  SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION                     I  3355
1449  CA 51 14     JZ   X1451 JMP IF NULL OR ':', END OF INSTRUCTION, NO 3RD ARG TO PROCESS  T  3356
144C  CF           RST  1  } SYNTAX CHECK FOR COMMA BETWEEN 2ND & 3RD ARGUMENTS           ,  3357
144D  2C           ~~INR L~~ }                                                            A  3358
144E  CD 81 14     CALL X1481 EVALUATE 3RD ARG TO 1 BYTE VALUE IN ACC                     T  3359
1451  C1     1451> POP  B  PUT MASK BYTE (2ND ARG) IN REG B
1452  DB 00  1452> IN   H'00' INPUT STATUS FROM SELECTED PORT (ARG 1)
```

DB ',' X141B DB ')'

```
1454 AB            XRA  E  XOR BYTE FROM 3RD ARG (STATUS BIT ACTIVE LO IF ARG3 ≠ Ø)
1455 A0            ANA  B  AND THE MASK BYTE FROM ARG 2
1456 CA 52 14      JZ   X1452  LOOP UNTIL STATUS BIT IS READY
1459 C9            RET
145A EB      X145A XCHG      PUT PTR TO STRING TO BE EVALUATED IN HL (WAS PUT IN DE AT ØE3C)
145B CF            RST  1  } SYNTAX CHECK FOR ')'
145C 29              DB')'-$40 }
145D C1      X145D POP  B  REMOVE THE RETURN ADDR FROM THE STACK
145E D1            POP  D  FETCH 2ND ARG FROM STACK (PUT ON STACK AT ØE3D)
145F C5            PUSH B  PUT THE RETURN ADDR ON THE STACK
1460 43            MOV  B,E  PUT 1-BYTE VALUE FOR 2ND ARG IN B
1461 04            INR  B  } TEST 2ND ARG VALUE
1462 05            DCR  B  }
1463 C0            RNZ  RETURN IF 2ND ARG VALUE IS NOT Ø
1464 C3 EE 08      JMP  X08EE  IF 2ND ARG IS Ø, PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"
1467 D7      X1467 RST  2  SCAN STRING TO NXT NON-SPACE CHAR, THEN Ø
1468 CD 52 0C X1468 CALL X0C52  EVALUATE ARGUMENT STRING TO # VALUE IN HOLDING AREA
146B E5      X146B PUSH H  SAVE PTR TO INSTRUCTION ON STACK
146C CD 11 1C      CALL X1C11  CONVERT ARGUMENT VALUE TO INTEGER FORMAT (CINT), RETURN
146F EB            XCHG  PUT INTEGER VALUE OF ARGUMENT IN DE
1470 E1            POP  H  FETCH PTR TO INSTRUCTION FROM STACK
1471 7A            MOV  A,D  PUT HI-ORDER BYTE OF VALUE INTO ACC
1472 B7            ORA  A  SET FLAGS - TEST FOR SIGN OF VALUE OF ARGUMENT STRING
1473 C9            RET  END OF ARGUMENT PROCESSING
1474 CD 81 14 X1474 CALL X1481  EVALUATE FIRST ARGUMENT TO 1 BYTE INTEGER VALUE
1477 32 53 14      STA  X1453  IF 'WAIT' FUNCTION CALL, THIS BYTE IS THE STATUS PORT TO BE
147A 32 3F 14      STA  X143F  IF 'OUT' FUNCTION CALL, THIS BYTE IS THE OUTPUT PORT #
147D CF            RST  1  } SYNTAX CHECK FOR ',' BETWEEN ARGUMENTS
147E 2C              DB','-$40 }
147F 06 D7         MVI  B,H'D7  DUMMY RST 2
1481 CD 52 0C X1481 CALL X0C52  EVALUATE ARGUMENT STRING TO # VALUE
1484 CD 6B 14 X1484 CALL X146B  CONVERT ARGUMENT TO INTEGER FORMAT
1487 C2 EE 08      JNZ  X08EE  IF ARGUMENT ≥ 256 (HI-ORDER BYTE≠Ø) PRINT "ILLEGAL FCN
148A 2B            DCX  H  BACK-UP PTR TO INSTRUCTION
148B D7            RST  2  SCAN TO NEXT NON-SPACE CHARACTER IN INSTRUCTION
148C 7B            MOV  A,E  PUT 1 BYTE ARGUMENT VALUE IN ACC
148D C9            RET
148E CD A8 13      CALL X13A8  IF ARG STRING LAST STRING IN STRING AREA, REMOVE IT, ADDR OF CHAR COUNT BYTE
1491 CA 02 19      JZ   X1902  IF ZERO FLAG SET, STRING IS NULL STRING, JMP TO 1902 TO STORE Ø IN EXPONENT
1494 5F            MOV  E,A  PUT CHAR COUNT BYTE OF ARG STRING IN E, D IS Ø (see 13AC)
1495 23            INX  H  ADVANCE PTR TO STRING FORMULA TABLE
1496 7E            MOV  A,M  } PUT ADDR OF 1ST CHAR OF ARG STRING IN HL
1497 23            INX  H    }
1498 66            MOV  H,M  }
1499 6F            MOV  L,A  }
149A E5            PUSH H  PUT PTR TO 1ST CHAR OF ARG STRING ON STACK
149B 19            DAD  D  COMPUTE ADDR OF 1ST CHAR OF STRING AFTER ARG STRING
149C 46            MOV  B,M  PUT 1ST CHAR OF NEXT STRING IN B
149D 72            MOV  M,D  PUT A NULL AT 1ST CHAR POSITION IN STRING AFTER ARG STRING
149E E3            XTHL  PUT PTR TO 1ST CHAR OF ARG STRING IN HL, PUT PTR TO NULL AFTER ARG STRING ON
149F C5            PUSH B  SAVE 1ST CHAR OF NEXT STRING ON STACK
14A0 7E            MOV  A,M  PUT 1ST CHAR OF ARG STRING IN ACC
14A1 CD 43 20      CALL X2043  CONVERT ARG STRING TO A NUMERIC VALUE IN HOLDING AREA
14A4 C1            POP  B  REMOVE 1ST CHAR OF NEXT-STRING FROM STACK
14A5 E1            POP  H  PUT PTR TO 1ST CHAR OF NEXT-STRING IN HL
14A6 70            MOV  M,B  PUT 1ST CHAR OF NEXT-STRING BACK WHERE IT BELONGS
14A7 C9            RET
14A8 C1            POP  B  REMOVE CNTL-C-CHK ADDR FROM STACK
14A9 CD 50 05      CALL X0550  set argument(s) in stack, converting line #'s to 2 bytes, BC =
14AC C5            PUSH B  save start addr of first line to list
```

```
3360
3361
3362
3363
3364.
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378  WITH INTEGER VALUE IN HL
3379
3380  EVALUATE STRING TO OBTAIN
3381  ARITHMETIC VALUE OF ARGUMENT
3382  CONVERT TO INTEGER FORMAT
3383
3384  1474 - EVALUATE 2 ARGUMENTS
3385       SEPARATED BY A COMMA
3386       TO 1 BYTE # VALUES
3387       2ND ARG IN ACC ON RETURN
3388
3389
3390  1481 - EVALUATE STRING TO VALUE
3391  1484 - CONVERT VALUE TO INTEGER
3392       ARGUMENT RANGE Ø - 255
3393       ONLY
3394       ARG IN ACC ON RETURN
3395
3396
3397  VAL
3398  - WHAT ABOUT TYPE CODE
3399
3400
3401  CONVERT ARG STRING TO A NUMERIC
3402  VALUE IN HOLDING AREA
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417  LIST
3418  of first byte in stmt area, DE = 1st line #
3419
```

JR
I
THEN
BCALL
W
MR
M
2SREAD
2?
O
,
W
MR
M
BCALL
+
W
I
M (IN HL)
BYTE
-
#
#
F
E
MC
A
M
E

89

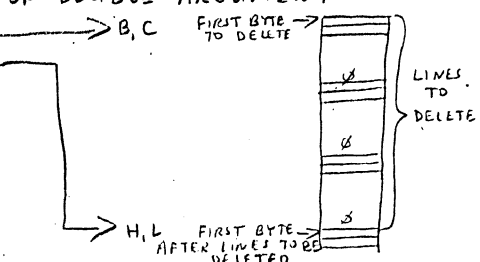
```
14AD  E1           X14AD  POP  H  remove addr of first byte of nxt stmt to list from stack   3420
14AF  D1                  POP  D  put line # of last line to list in DE                      Q    3421
14AF  4E                  MOV  C,M } FETCH CHAIN FROM STMT POINTED TO BY HL                 N    3422
14B0  23                  INX  H   } STORE ADDR OF NXT STMT (CHAIN ADDR) IN BC              #    3423
14B1  46                  MOV  B,M }                                                        F    3424
14B2  23                  INX  H  POINT TO LINE # OF STMT                                   #    3425
14B3  78                  MOV  A,B } TEST FOR DOUBLE NULL (END OF STMT AREA)                     3426
14B4  B1                  ORA  C   }                                                        1    3427
14B5  CA 0E 04            JZ   X040E JMP TO ROUTINE TO PRINT "OK" - RETURN TO BASIC        J^   3428 COMMAND MODE
14B8  CD EB 07            CALL X07EB TEST FOR CNTL-C HAVING BEEN TYPED, RETURN HERE IF     M    3429 NO CHAR TYPED
14BB  C5                  PUSH B SAVE FIRST-BYTE ADDRESS OF NXT STMT                       E    3430
14BC  4E                  MOV  C,M } FETCH LINE # FROM STMT POINTED TO BY H,L              N    3431
14BD  23                  INX  H   } STORE IN BC                                           #    3432
14BE  46                  MOV  B,M }                                                       F    3433
14BF  23                  INX  H  POINT TO ASCII AREA OF STMT                              #    3434
14C0  C5                  PUSH B  SAVE CURRENT LINE # ON STACK (LINE # of current stmt being E   3435 listed)
14C1  E3                  XTHL SAVE PTR TO STMT ON STACK, PUT CURRENT LINE # IN HL                3436
14C2  EB                  XCHG AFTER XCHG, DE = CURRENT LINE #        HL = LINE # OF LAST STMT 3437 TO LIST.
14C3  E7                  RST  4  RC IF HL < DE (ie present line is not to be listed)      3438
14C4  C1                  POP  B  BC IS PTR TO STMT AREA (ASCII PART OF STMT TO LIST)       A    3439
14C5  DA DD 04            JC   X04DD NO MORE LINES TO LIST, PRINT "OK" + RETURN TO TELETYPE Z}   3440 COMMAND MODE
14C8  E3                  XTHL PUT LAST LINE # TO LIST BACK ON STACK, PUT FIRST-BYTE ADDRESS OF NXT 3441 STMT IN HL
14C9  E5                  PUSH H PUT FIRST-BYTE ADDR OF NXT STMT ON STACK                  3442
14CA  C5                  PUSH B PUT PTR TO STMT ON STACK                                  E    3443
14CB  EB                  XCHG AFTER XCHG, DE = FIRST BYTE ADDRESS OF NXT STMT, HL = LINE # OF    3444 PRESENT LINE
14CC  CD 98 0A            CALL X0A98 PRINT CR LF & NULLS, BEFORE 1ST LINE & AFTER EVERY LINE M    3445
14CF  CD 73 21            CALL X2173 CONVERT LINE # TO DECIMAL ASCII FORM & PRINT           H !  3446
14D2  3E 20               MVI  A,A' ' } PRINT SPACE AFTER LINE #                           >    3447
14D4  E1                  POP  H       } HL = PTR TO STMT (ASCII AREA)                          3448
14D5  DF                  RST  3       }                                                   _    3449
14D6  CD E9 14            CALL X14E9 LOAD LINE BUFR WITH EXPANDED STMT FROM STMT AREA      M    3450
14D9  21 5A 03            LXI  H,X035A LOAD HL WITH ADDR OF FIRST BYTE OF THE LINE BUFR    !Z   3451
14DC  01 AD 14            LXI  B,X14AD } PUT THE RETURN ADDR 14AD ON THE STACK FOR USE     -    3452
14DF  C5                  PUSH B       } BY A RETURN IN SUBROUTINE 1244                    E    3453
14E0  2B           X14E0  DCX  H  DECREMENT PTR TO LINE BUFR                               +    3454
14F1  06 00               MVI  B,H'00' SET TERMINATOR SEARCH CHAR TO BE A NULL                  3455
14E3  CD FF 11            CALL X11FF GENERATE A CHAR COUNT BYTE BY COUNTING CHAR'S IN THE STRING TILM A  3456 NULL FOUND
14E6  C3 44 12  implied jmp JMP X1244 PRINT THE CHAR STRING                                CD   3457
-------------------------------------------------------------------------------------------------------------
14E9  01 59 03     X14E9  LXI  B,X0359 LOAD B,C WITH ADDR OF BYTE BEFORE FIRST BYTE OF LINE Y  3458 BUFR
14EC  16 E1 POP H         MVI  D,H'E1' DUMMY BYTE TO SKIP POP H TO ENTER LOOP [POP H RESTORES PTR 3459 TO STMT IN HL]
14EE  7E           X14EE  MOV  A,M FETCH A BYTE INTO ACC FROM ASCII AREA OF STMT BEING LISTED  3460
14EF  03                  INX  B ADVANCE PTR TO LINE BUFR ; POINT TO NXT BYTE AVAILABLE     3461
14F0  B7                  ORA  A TEST BYTE FOR NULL                                        7    3462
14F1  23                  INX  H ADVANCE PTR TO ASCII PART OF STMT                         #    3463
14F2  02                  STAX B STORE CHAR IN LINE BUFR                                        3464
14F3  C8                  RZ   NULL HAS BEEN FOUND - EXIT                                  H    3465
14F4  F2 EE 14            JP   X14EE IF NOT A TOKEN, REPEAT LOOP FOR NXT CHAR IN STMT           3466
14F7  FE 93               CPI  H'93' IS TOKEN BYTE = "ELSE"                                     3467
14F9  CC 9B 1C            CZ   X1C9B → DCX B RET  REMOVES COLON IN FRONT OF ELSE (INSERTED  L8   3468 WHILE CONDENSING INPUT BUFFER)
14FC  D6 7F               SUI  H'7F' CONVERT TOKEN TO OFFSET (# OF RESERVED WRD IN LIST)    V    3469
14FE  E5                  PUSH H SAVE PTR TO STMT IN STACK                                      3470
14FF  11 7A 00            LXI  D,X007A LOAD DE WITH ADDR OF FIRST BYTE OF RESERVED WRD TABLE     3471
1502  D5           X1502  PUSH D SAVE ADDR OF FIRST BYTE OF RESERVED WRD BEING SCANNED      U    3472
1503  F5                  PUSH PSW SAVE RESERVED WRD COUNT IN STACK (ADJUSTED TOKEN)             3473
1504  1A           X1504  LDAX D FETCH A BYTE OF RESERVED WRD                                   3474
1505  13                  INX  D ADVANCE PTR TO RESERVED WRD                                    3475
1506  B7                  ORA  A TEST FOR D7 SET (INDICATES BYTE IS LAST CHAR IN WRD)  7    3476
1507  F2 04 15            JP   X1504 LOOP IF END OF WRD NOT FOUND YET                           3477
150A  F1                  POP  PSW } AFTER END OF WRD FOUND, FETCH WRD CTR FROM STACK            3478
150B  3D                  DCR  A   } DECR COUNT BY 1                                       =    3479
```

```
CHN ADDR
LINE #
ASCII
AREA
OF
STMT
Ø
```

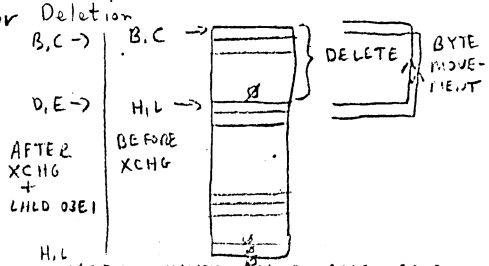
```
150C  E1              POP  H PUT ADDR OF 1ST BYTE OF RESERVED WRD IN HL
150D  C2 02 15        JNZ  X1502 LOOP UNTIL WRD COUNT IS Ø
1510  7E        X1510 MOV  A,M FETCH A BYTE FROM RESERVED WRD TABLE
1511  B7              ORA  A  TEST FOR D7 SET (LAST CHAR OF WRD)
1512  02              STAX B  STORE CHAR IN LINE BUFR
1513  FA ED 14        JM   X14ED JMP IF LAST CHAR IN RESERVED HAS BEEN MOVED
1516  03              INX  B } ADVANCE PTR TO LINE BUFR
1517  23              INX  H } ADVANCE PTR TO RESERVED WRD TABLE
1518  C3 10 15        JMP  X1510 LOOP TO MOVE NXT CHAR
```

DELETE

```
1518  CD 50 05        CALL X0550 set argument(s) in stack converting line #'s to 2 bytes
151E  D1              POP  D  FETCH LINE # (IF SINGLE ARGUMENT) or 2ND LINE # (IF 2
151F  C5              PUSH B  SAVE ADDR OF FIRST BYTE OF FIRST LINE TO BE DELETED
1520  CD 70 05        CALL X0570 } IF NO LINE # TO MATCH SINGLE ARGUMENT OR
1523  D2 EE 08        JNC  X08EE } PRINT 'ILLEGAL FUNCTION CALL'
1526  54              MOV  D,H } HL PT TO FIRST BYTE OF LN AFTER LINES TO BE DELETED
1527  5D              MOV  E,L } SAVE THIS ADDR IN DE
1528  E3              XTHL    } AFTER THESE INSTR  STACK
1529  E5              PUSH H  }  ADDR OF 1ST BYTE OF LINE AFTER DELETED LINES } HL = FIRST
152A  E7              RST  4  } TESTS FOR RELATIVE SIZE OF DOUBLE ARGUMENTS (RC IF HL < DE)  ADDR OF 1ST BYTE TO BE DELETED } DE = LAST
152B  D2 EE 08        JNC  X08EE } ie DELETE 20,10 IS ILLEGAL, ERROR ILLEGAL FUNCTION CALL
152E  21 48 04        LXI  H,X0448 LOC OF " CR LF OK CR LF
1531  CD 41 12        CALL X1241  PRINT ASCII STRING
1534  C1              POP  B BC = FIRST BYTE TO BE DELETED
1535  21 40 05        LXI  H,X0540 } IF ENTERING X1539 DELETE ROUTINE FROM ABOVE, SET RETURN ADDR
1538  E3              XTHL         } Ø54Ø ONTO THE STACK. Ø54Ø DOES A 'NEW' + RESETS THE CHAIN ADDRESSES
                                     HL = ADDR OF BYTE AFTER LAST BYTE TO BE DELETED
1539  EB        X1539 XCHG  D,E POINT TO BYTE TO MOVE AFTER XCHG
153A  2A E1 03        LHLD X03E1  H,L POINT TO NXT AVAILABLE BYTE IN STMT AREA
153D  1A        X153D LDAX D  } MOVE 1 BYTE
153E  02              STAX B  }
153F  03              INX  B  } ADJUST POINTERS (SOURCE + DESTINATION) TO MOVE NXT BYTE
1540  13              INX  D  }
1541  F7              RST  4  MOVE ANOTHER BYTE (DOES H,L = D,E) ?
1542  C2 3D 15        JNZ  X153D YES
1545  60              MOV  H,B }
1546  69              MOV  L,C }
1547  22 E1 03        SHLD X03E1 } ADJUST PTR FOR NXT AVAILABLE BYTE
154A  C9              RET
```



Stmt Area Byte Movement For Deletion



Ø3E1, Ø3E2 NXT BYTE AVAILABLE IN STMT AREA

PEEK

```
154B  CD 11 1C        CALL X1C11 CONVERT VALUE IN HOLDING AREA TO INTEGER FORMAT, PUT IN HL
154E  7E              MOV  A,M  FETCH SPECIFIED BYTE FROM MEMORY INTO ACC
154F  C3 04 10        JMP  X1004 CONVERT BYTE IN ACC TO 2-BYTE INTEGER FORMAT STORED IN HOLDING AREA
```

POKE

```
1552  CD 68 14        CALL X1468 CONVERT 1ST ARG TO 2-BYTE INTEGER IN DE
1555  D5              PUSH D  SAVE 1ST ARG VALUE ON STACK
1556  CF              RST  1  } SYNTAX CHECK FOR ',' BETWEEN 1ST + 2ND ARGUMENTS
1557  2C        DB',' INR  L  }
1558  CD A1 14        CALL X14A1 EVALUATE 2ND ARGUMENT TO 1 BYTE INTEGER VALUE IN ACC
155B  D1              POP  D  FETCH ADDR TO POKE (1ST ARG) FROM STACK
155C  12              STAX D  STORE BYTE (POKE) AT SPECIFIED ADDR
155D  C9              RET
```

FORMATTED OUTPUT ROUTINE

USING

```
155E  CC 53 0C  X155E CALL XGC53 EVALUATE STRING VALUE (CHAR'S + VAR'S OK) - 'USING' FORMAT STRING
1561  CD 88 1C        CALL X1C88 TEST FORMAT VALUE, MUST BE STRING TYPE, OTHERWISE - "TYPE MISMATCH"
1564  CF              RST  1  } SYNTAX CHECK FOR ';' BETWEEN 'USING' FORMAT STRING + PRINT
1565  3B        DB';' DCX  SP } STMT ARGUMENTS
1566  EB              XCHG PUT TO INSTRUCTION STRING IN DE
1567  2A 12 04        LHLD XG412 LOAD HL WITH ADDR OF CHAR COUNT BYTE (IDENTITY GROUP) OF FORMAT STRING
156A  01 01 EB  POP D XCHG LXI  D,XEB01 DUMMY AFTER BOTH INSTRUCTIONS - PTR TO IDENTITY GROUP IN HL -
156D  E5              PUSH H PUT PTR TO IDENTITY GROUP OF FORMAT STRING ON STACK
156E  AF              XRA  A  CLEAR ACC
156F  BA              CMP  D TEST HI-BYTE OF PTR TO INSTRUCTION - BYTE WILL BE ØØ IF INSTRUCTION IS IMMEDIATE
1570  F5              PUSH PSW SAVE RESULT ON STACK
1571  D5              PUSH D SAVE PTR TO INSTRUCTION ON STACK
```

(FOUND BY 155E - CALL ØC53)

LOCATED IN LINE BUFR Ø35A - Ø3A1

```
1572  46           MOV  B,M  } PUT CHAR COUNT BYTE OF FORMAT STRING IN B & TEST IT                    F    3540
1573  B0           ORA  B    }                                                                        0    3541
1574  CA EE 08     JZ   X08EE IF FORMAT STRING IS THE NULL STRING, PRINT ERROR MESSAGE "ILLEGAL     J    3542 FUNCTION CALL"
1577  23           INX  H    }                                                                        #    3543
1578  4E           MOV  C,M  } PUT ADDR OF 1ST CHAR OF FORMAT STRING IN HL                            N    3544
1579  23           INX  H    }                                                                        #    3545
157A  66           MOV  H,M  }                                                                             3546
157B  69           MOV  L,C  }                                                                             3547
157C  C3 9C 15     JMP  X159C JMP INTO FORMAT PROCESSOR ROUTINE                                       C    3548
------------------------------------------------------------------------------------------------------------------
157F  58     X157F MOV  E,B SAVE FORMAT STRING CHAR COUNT IN E  } IF 2ND '\' NOT FOUND               X    3549
1580  E5           PUSH H  SAVE PTR TO FORMAT STRING ON STACK (1ST '\' +1) } BEFORE NXT NON-SPACE CHAR, 1ST '\'  3550
1581  0E 02        MVI  C,H'02' SET STRING FIELD COUNT TO 2 (ONE      FOR EACH '\'   NOT A FORMAT CHAR  3551
1583  7E     X1583 MOV  A,M FETCH NXT FORMAT CHAR INTO ACC                                                 3552
1584  23           INX  H   ADVANCE PTR TO NXT FORMAT CHAR                                            #    3553
1585  FE 5C        CPI  A'\' TEST FORMAT CHAR                                                         \    3554
1587  CA C2 16     JZ   X16C2 IF FORMAT CHAR IS '\' - 2ND '\' - PRINT                                 JB   3555
158A  FE 20        CPI  A' ' TEST FORMAT CHAR                                                         .    3556
158C  C2 94 15     JNZ  X1594 IF CHAR NOT A ' ' - THEN 1ST '\' NOT TERMINATED BY A 2ND '\'            B    3557 MUST NOT BE A FORMAT CHAR
158F  0C           INR  C   INCR. STRING FIELD COUNT                                                       3558
1590  05           DCR  B DECREMENT FORMAT STRING CHAR COUNT                                               3559
1591  C2 83 15     JNZ  X1583 IF FORMAT STRING NOT USED-UP YET, TRY NXT CHAR. IF COUNT B IS Ø          B    3560, 1ST '\' NOT TERMINATED PROPERLY
1594  E1     X1594 POP  H RESTORE HL TO POINT TO CHAR AFTER '\'                                            3561
1595  41           MOV  B,F RESTORE FORMAT STRING CHAR COUNT TO ITS FORMER VALUE (see 157F)                3562
1596  3E 5C        MVI  A,A'\' PUT '\' IN ACC (1ST '\')                                               >\   3563
1598  CD F4 16     CALL X16F4 PRINT LEADING '+' IF D≠Ø (+ NOT A FORMAT CHAR IN THIS CASE) ACC NOT CHANGED  M    3564
159B  CF           RST  3   PRINT CHAR IN ACC                                                         _    3565
159C  AF     X159C XRA  A  CLEAR ACC                                                                 /    3566 START A NEW NUMERIC FIELD
159D  5F           MOV  E,A CLEAR CTR THAT COUNTS NUMERIC FIELD POSITIONS TO LEFT OF DECIMAL PT       _    3567
159E  57           MOV  D,A CLEAR BIT-FLAG BYTE (USED FOR '+' '$$' '**' '*$')                         W    3568
159F  CD F4 16     CALL X16F4 CHECK D (BIT-FLAG BYTE) - IF D≠Ø, PREVIOUS '+' NOT A FORMAT CHAR, SO PRINT '+'  M  3569
15A2  57           MOV  D,A  PUT NEW VALUE FOR BIT-FLAG IN D                                          W    3570
15A3  7E           MOV  A,M  FETCH A CHAR FROM FORMAT STRING INTO ACC                                      3571
15A4  23           INX  H   ADVANCE PTR TO NXT CHAR IN FORMAT STRING                                  #    3572
15A5  FE 21        CPI  A'!' TEST FORMAT CHAR                                                         !    3573
15A7  CA 9F 16     JZ   X169F IF FORMAT CHAR IS '!' - PRINT A SINGLE CHAR STRING FIELD                J?   3574
15AA  FE 23        CPI  A'#' TEST FORMAT CHAR                                                         #    3575
15AC  CA EE 15     JZ   X15EE JMP TO 15EE TO COUNT NUMERIC-FIELD POSITIONS                            J    3576
15AF  05           DCR  B DECREMENT FORMAT STRING CHAR COUNT                                               3577
1580  CA AD 16     JZ   X16AD IF FORMAT STRING USED-UP, CHECK FOR SIGN CHAR, AND REPEAT USING ENTIRE FORMAT  J-  3578 STRING IF NECESSARY
1583  FE 2B        CPI  A'+' TEST FORMAT CHAR                                                         +    3579
1585  3E 08        MVI  A,H'08' PUT BIT CODE FOR '+' IN ACC                                           >    3580
1587  CA 9F 15     JZ   X159F JMP TO 15EE TO PRINT ANY PREVIOUS '+' CHAR                              J    3581
158A  2B           DCX  H  BACK-UP PTR } FETCH FORMAT CHAR INTO ACC AGAIN                              +    3582
158B  7E           MOV  A,M FETCH CHAR } HL PT TO NXT FORMAT CHAR                                          3583
158C  23           INX  H ADVANCE PTR  }                                                              #    3584
158D  FE 2E        CPI  A'.' TEST FORMAT CHAR                                                         .    3585
158F  CA 0D 16     JZ   X160D JMP TO 160D TO HANDLE '.' AS 1ST FORMAT CHAR FOR A NUMERIC FIELD        J    3586
1592  FE 5C        CPI  A'\' TEST FORMAT CHAR                                                         \    3587
1594  CA 7F 15     JZ   X157F IF FORMAT CHAR IS '\' - PRINT N-CHAR STRING FIELD OF NEXT ARG STRING VALUE  JAN  3588
1597  0E           CMP  M TEST CURRENT FORMAT CHAR AGAINST NEXT FORMAT CHAR                           >    3589
1598  C2 98 15     JNZ  X1598 JMP IF FORMAT CHAR'S ARE NOT THE SAME                                   B    3590 NEXT FORMAT CHAR'S MUST BE $$ ** or **$
159B  FE 24        CPI  A'$' TEST FORMAT CHAR PAIR                                                    $    3591
159D  CA A7 15     JZ   X15E7 JMP IF PAIR IS '$$'                                                     J    3592
15A0  FE 2A        CPI  A'*' TEST FORMAT CHAR PAIR                                                    *    3593
15A2  C2 98 15     JNZ  X1598 JMP IF PAIR IS NOT '**'                                                 B    3594
15A5  78           MOV  A,B PUT FORMAT STRING CHAR COUNT IN ACC                                            3595 '**'
15A6  FE 02        CPI  H'02' TEST FORMAT STRING CHAR COUNT                                                3596
15A8  23           INX  H ADVANCE PTR TO NXT CHAR IN FORMAT STRING                                    #    3597
15A9  CA CF 15     JC   X15CF JMP IF LAST CHAR'S IN FORMAT STRING ARE '**' (1ST '*' WAS COUNTED AT 15AF)  Z  3598
15AC  7E           MOV  A,M PUT NXT FORMAT CHAR IN ACC (CHAR AFTER '**')                                   3599
```

```
15D0  FE 24        CPI   A'$' TEST FORMAT CHAR                          $   3600
15DF  3E 20        X150F MVI   A,A' ' SET 2Ø BIT IN ACC - BIT FLAG FOR '*' FILL   >   3601
15E1  C2 E8 15     '**' or '***' JNZ   X15E8 JMP IF PROCESSING '**' or NOT PROCESSING '***$'   8   3602
15E4  05                 DCR   B DECREMENT FORMAT STRING CHAR COUNT (COUNTS 3RD CHAR - '$')   3603 '***$'
15E5  1C                 INR   E ADD 1 TO NUMERIC-FIELD CTR (FOR $)          3604
15E6  FE AF        XRA A CPI   H'AF' DUMMY  '$$'  '**'  '***$'  - SET 1Ø BIT IN ACC - BIT FLAG   /   3605
15E8  C6 10              ADI   H'10' RESULT IN ACC  1Ø  2Ø  3Ø                FOR '$' IMMEDIATELY BEFORE # TO PRINT   3606
15EA  23                 INX   H ADVANCE PTR TO NXT CHAR IN FORMAT STRING (CHAR AFTER '***$' or '$$')   #   3607
15EB  1C           X15EB INR   E ADD 1 TO NUMERIC-FIELD CTR (1ST $ OF '$$' or 1ST '*' OF '**' or '***$')   3608
15EC  82                 ADD   D  '$$'  '**'  '***$'                         3609
15ED  57                 MOV   D,A RESULT IN D  1Ø+(D)  2Ø+(D)  3Ø+(D)      W   3610 or FOUND THEREAFTER)
15EE  1C           X15EE INR   E ADD 1 TO NUMERIC-FIELD CTR (2ND CHAR OF '$$', '**', or '***$' or each # FOUND THEREAFTER)   3611
15EF  0E 00              MVI   C,H'00' ZERO THE DECIMAL-PT POSITION CTR      3612
15F1  05                 DCR   B DECREMENT FORMAT STRING CHAR COUNT (COUNTS 2ND CHAR OF '$$', '**', or '***' or COUNTS FORMAT CHAR   3613 - COUNTS LAST CHAR OF PREFIX FORMAT GROUP)
15F2  CA 43 16           JZ    X1643 JMP IF FORMAT STRING USED-UP —  JC FOUND   3614 AT 15EC('#') or AT 1626(',')
15F5  7E                 MOV   A,M PUT NXT FORMAT CHAR IN ACC                3615
15F6  23                 INX   H  ADVANCE PTR TO NXT CHAR IN FORMAT STRING   #   3616
15F7  FE 2E              CPI   A'.' TEST FORMAT CHAR                         .   3617
15F9  CA 18 16           JZ    X1618 JMP IF FORMAT CHAR IS '.' —             J   3618
15FC  FE 23              CPI   A'#' TEST FORMAT CHAR                         #   3619
15FE  CA EE 15           JZ    X15EE JMP IF FORMAT CHAR IS '#' - ADD 1 COUNT TO NUMERIC-FIELD CTR   J   3620
1601  FE 2C              CPI   A',' TEST FORMAT CHAR                         ,   3621
1603  C2 24 16           JNZ   X1624 JMP IF FORMAT CHAR IS NOT A ',' —       8$  3622
1606  7A                 MOV   A,D } SET 4Ø BIT OF BYTE IN D REG TO INDICATE ',' FOUND ON LEFT SIDE   3623 ','
1607  F6 40              ORI   A'@' } OF '.', IF ANY                         @   3624
1609  57                 MOV   D,A                                           H   3625
160A  C3 EE 15           JMP   X15EE CONTINUE TESTING NEXT FORMAT CHAR       C   3626
                                                                                  3627 '.' - LEADING '.'
1600  7E           X1600 MOV   A,M  PUT NEXT FORMAT CHAR IN ACC
160E  FE 23              CPI   A'#' TEST FORMAT CHAR                         #   3628
1610  3E 2E              MVI   A,A'.' } PUT '.' IN ACC, IF FORMAT CHAR AFTER '.' IS NOT A '#', THEN '.'   >.  3629
1612  C2 98 15           JNZ   X1598 } IS NOT A FORMAT CHAR - PRINT '.' & CONTINUE PROCESSING AT START OF NXT   @   3630 FIELD
1615  0E 01              MVI   C,H'01' INITIALIZE DECIMAL-PT POSITION CTR TO 1 (1 PRINT POSITION FOR '.' CHAR)   3631 '.' AS FORMAT CHAR
1617  23                 INX   H ADVANCE PTR TO NXT FORMAT CHAR              #   3632
1618  0C           X1618 INR   C ADD 1 TO DECIMAL-PT POSITION CTR                3633
1619  05                 DCR   B DECREMENT FORMAT STRING CHAR COUNT (COUNTS '.' FOUND AT 15BD or 15F7 or '#' FOUND   3634 AT 161F
161A  CA 43 16           JZ    X1643 JMP IF FORMAT STRING USED-UP            JC  3635
161D  7E                 MOV   A,M PUT NXT FORMAT CHAR IN ACC                3636
161E  23                 INX   H ADVANCE PTR TO NXT CHAR IN FORMAT STRING    #   3637
161F  FE 23              CPI   A'#' TEST FORMAT CHAR                         #   3638
1621  CA 18 16           JZ    X1618 JMP IF FORMAT CHAR IS '.' — ADD 1 DECIMAL PT POSITION CTR, DECR FORMAT STRING CHAR COUNT, CONTINUE LOOPING TIL NON-FORMAT   J   3639
1624  D5           X1624 PUSH  D SAVE DE ON STACK                            U   3640 CHAR FOUND
1625  11 41 16           LXI   D,X1641 } PUT ADDR 1641 ON STACK FOR IMPLIED JMP (SAVES MEMORY - 1 BYTE FOR   A   3641 CONDITIONAL RETURNS)
1628  D5                 PUSH  D  }                                          U   3642
1629  54                 MOV   D,H } SAVE PTR TO FORMAT STRING IN DE (IN CASE 4 '↑' IN A ROW NOT FOUND   T   3643
162A  5D                 MOV   E,L }                                         T   3644
162B  FE 5E              CPI   A'↑' TEST FORMAT CHAR                         ^   3645
162D  C0                 RNZ   IMPLIED JMP TO 1641 IF FORMAT CHAR NOT '↑'    @   3646
162E  BE                 CMP   M TEST 2ND FORMAT CHAR                        >   3647 1 '↑' FOUND
162F  C0                 RNZ   IMPLIED JMP TO 1641 IF 2ND FORMAT CHAR NOT '↑'   @   3648
1630  23                 INX   H ADVANCE PTR TO FORMAT STRING                #   3649
1631  BE                 CMP   M TEST 3RD FORMAT CHAR                        >   3650 2ND '↑' FOUND
1632  C0                 RNZ   IMPLIED JMP TO 1641 IF 3RD FORMAT CHAR NOT '↑'   @   3651
1633  23                 INX   H ADVANCE PTR TO FORMAT STRING                #   3652
1634  BE                 CMP   M TEST 4TH FORMAT CHAR                        >   3653 3RD '↑' FOUND
1635  C0                 RNZ   IMPLIED JMP TO 1641 IF 4TH FORMAT CHAR NOT '↑'   @   3654
1636  23                 INX   H ADVANCE PTR TO NXT CHAR IN FORMAT STRING (CHAR AFTER '↑↑↑↑')   #   3655 4TH '↑' FOUND
1637  7B                 MOV   A,B PUT FORMAT STRING CHAR COUNT IN ACC               3656
1638  D6 04              SUI   H'04' SUBTRACT 4 FROM FORMAT STRING CHAR COUNT (FOR '↑↑↑↑')   V   3657
163A  D8                 RC    IMPLIED JMP TO 1641 IF CHAR COUNT WAS 3 OR LESS   X   3658
163B  D1                 POP   D REMOVE IMPLIED JMP ADDR 1641 FROM STACK     Q   3659
```

```
##,##
**,##
**$.##
$$.##
 E  C
```

```
163C  D1                  POP  D  RESTORE DE FROM STACK (see 1624)
163D  47                  MOV  B,A PUT ADJUSTED FORMAT STRING CHAR COUNT BACK INTO REG B
163E  14                  INR  D SET Ø1 BIT OF BIT-FLAG BYTE TO INDICATE EXPONENTIAL FORMAT ('↑↑↑↑' FOUND IN
163F  23                  INX  H ADVANCE PTR TO FORMAT CHAR STRING (TO MATCH INX H AT 1636, 1633, 163Ø, 162E, 15F6)
1640  CA EB 01  XCHG POP D JZ   XØ1EB DUMMY RESTORE PTR TO FORMAT STRING TO WHERE IT WAS (see 1629-162A), put it in HL, restore
1643  7A          X1643   MOV  A,D PUT BIT-FLAG BYTE IN ACC
1644  2B                  DCX  H BACK-UP PTR TO LAST FORMAT CHAR TESTED
1645  1C                  INR  E ADD 1 TO NUMERIC FIELD POSITION (THIS IS FOR LEADING '+' IF D ≠ Ø)
1646  E6 08               ANI  H'08' TEST BIT FLAG FOR '+' IN FRONT OF NUMERIC FIELD (see 1583)
1648  C2 63 16            JNZ  X1663 JMP IF LEADING '+' FOUND AT 1583 (FLAG BIT Ø8 SET)
164B  1D                  DCR  E REMOVE 1 NUMERIC FIELD COUNT (CANCEL 1645 - NO LEADING SIGN CHAR)
164C  78                  MOV  A,B PUT FORMAT STRING CHAR COUNT IN ACC
164D  B7                  ORA  A   TEST CHAR COUNT
164F  CA 63 16            JZ   X1663 JMP IF FORMAT STRING USED-UP
1651  7E                  MOV  A,M PUT LAST FORMAT CHAR TESTED IN ACC AGAIN
1652  D6 2D               SUI  A'-' TEST FORMAT CHAR
1654  CA 5E 16            JZ   X165E JMP IF FORMAT CHAR IS TRAILING '-'
1657  FE FE               CPI  H'FE' TEST FORMAT CHAR ('+' CHAR IS FE AT THIS POINT)
1659  C2 63 16            JNZ  X1663 JMP IF FORMAT CHAR IS NOT TRAILING '+'
165C  3E 08               MVI  A,H'08' SET Ø8 BIT FOR '+' IN BIT-FLAG BYTE
165E  C6 04       X165E   ADI  H'04'  SET Ø4 BIT FOR TRAILING SIGN CHAR IN BIT FLAG BYTE
1660  82                  ADD  D COMBINE THESE NEW BITS WITH FORMER BIT-FLAG BYTE
1661  57                  MOV  D,A STORE RESULT IN BIT-FLAG REGISTER
1662  05                  DCR  B DECREMENT FORMAT STRING CHAR COUNT (COUNTS TRAILING '+' or '-')
1663  E1        X1663     POP  H PUT PTR TO INSTRUCTION IN HL (see 1571)
1664  F1                  POP  PSW FETCH RESULT-OF-TEST (see 156F) STATUS FROM STACK
1665  CA B6 16            JZ   X16B6 JMP IF PRINT USING INSTRUCTION IS IMMEDIATE (NOT IN A BASIC PROGRAM)
1668  C5                  PUSH B SAVE BC ON STACK
1669  D5                  PUSH D SAVE DE ON STACK
166A  CD 52 0C            CALL X0C52 EVALUATE NEXT ARG OF PRINT STMT TO VALUE IN HOLDING AREA
166D  D1                  POP  D RESTORE DE FROM STACK
166E  C1                  POP  B RESTORE BC FROM STACK
166F  C5                  PUSH B SAVE REMAINING FORMAT STRING CHAR COUNT (B) & DECIMAL-PT POSITION CTR (C)
1670  E5                  PUSH H SAVE PTR TO INSTRUCTION ON STACK
1671  43                  MOV  B,E } PUT # OF NUMERIC FIELD POSITIONS TO LEFT OF DECIMAL PT IN B
1672  78                  MOV  A,D }     # OF NUMERIC FIELD POSITIONS TO RIGHT OF DP INCLUDING DP IN C
1673  81                  ADD  C   } COMPUTE TOTAL NUMERIC FIELD POSITIONS COUNT IN ACC
1674  FE 19               CPI  H'19' TEST NUMERIC FIELD COUNT
1676  D2 EE 08  ERROR     JNC  X08EE IF COUNT IS ≥ 25, PRINT ERROR MESSAGE 'ILLEGAL FUNCTION CALL' (DBL PREC
1679  7A                  MOV  A,D  PUT BIT-FLAG BYTE IN ACC
167A  F6 80               ORI  H'80' SET 8Ø BIT OF BIT-FLAG BYTE FOR #-CONVERT ROUTINE
167C  CD 82 21            CALL X2182 CONVERT ARG-TO-PRINT TO A CHAR STRING
167F  CD 41 12            CALL X1241 PRINT CONVERTED-ARG CHAR STRING (COUNTS CHAR'S, REMOVE STRING FROM
1682  E1          X1682   POP  H PUT PTR TO INSTRUCTION IN HL
1683  2E                  DCX  H BACK-UP PTR
1684  D7                  RST  2 SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION  -RZ IF NULL OR COLON FOUND
1685  37                  STC    SET CARRY - IF END OF INSTRUCTION, FORCES PRINTING CRLF AT 1686
1686  CA 94 16            JZ   X1694 JMP IF CHAR IN INSTRUCTION IS A NULL OR COLON
1689  FE 3B               CPI  A':' TEST CHAR
168B  CA 93 16            JZ   X1693 JMP IF DELIMITER BETWEEN ARGS-TO-PRINT IS A ';' - IGNORE IT
168E  FE 2C               CPI  A',' TEST CHAR
1690  C2 A9 04  ERROR     JNZ  X04A9 IF DELIMITER IS NOT A ',' - PRINT ERROR MESSAGE "SYNTAX ERROR" - OTHERWISE
1693  D7          X1693   RST  2 SCAN TO NXT NON-SPACE CHAR IN INSTRUCTION
1694  C1          X1694   POP  B PUT REMAINING FORMAT STRING CHAR COUNT IN B + DON'T CARE ABOUT C
1695  EB                  XCHG PUT PTR TO INSTRUCTION IN DE
1696  E1                  POP  H } PUT PTR TO IDENTITY GROUP OF FORMAT STRING IN HL (see 1567-156D)
1697  E5                  PUSH H } LEAVE IT ON STACK
1698  F5                  PUSH PSW SAVE FLAG STATUS FROM RST 2 AT 1693 ON STACK
1699  D5                  PUSH D PUT PTR TO INSTRUCTION ON STACK
169A  7E                  MOV  A,M PUT ORIGINAL CHAR COUNT BYTE OF FORMAT STRING IN ACC
```

D IS BIT-FLAG BYTE E IS NUMERIC-FIELD CTR TO LEFT OF DP

```
3660
3661
3662 FORMAT STRING)
3663
3664 DE FROM STACK (see 1624)
3665
3666  NOTE: LEADING SIGN CHAR IS COUNTED
3667   IN VALUE IN E
3668        TRAILING SIGN IS COUNTED IN
3669   NEITHER E or C, BUT IDENTIFIED
3670          BY Ø4 FLAG BIT
3671  TRAILING SIGN CHAR NOT COUNTED
3672
3673
3674
3675
3676 BIT-FLAG BYTE FORMAT
3677   Ø1 EXPONENTIAL FORMAT '↑↑↑↑'
3678   Ø2
3679   Ø4 TRAILING SIGN CHAR '+' or '-'
3680   Ø8 '+' = 1   '-' = Ø
3681   1Ø '$$'   3Ø = '**$'
3682   2Ø '**'
3683   4Ø COMMA FOUND TO LEFT OF '.'
3684   8Ø BIT FLAG FOR # CONVERSION = 1
3685
3686
3687
3688
3689
3690
3691
3692 STACK
3693
3694
3695
3696
3697                    1   1    4
3698 16 DIGITS + SIGN + DP + EXPONENT = 22 )
3699
3700
3701
3702 AREA, PRINTS)
3703
3704
3705 NON-SPACE CHAR IN ACC
3706
3707
3708
3709
3710
3711 IGNORE THE DELIMITER
3712
3713
3714
3715
3716
3717
3718
3719
```

```
1690 90          SUB  B SUBTRACT REMAINING COUNT OF FORMAT STRING FROM TOTAL CHAR COUNT (OFFSET 3720 TO COMPUTE ADDR OF 1ST REMAINING CHAR)
1691 23          INX  H                                                                  #    3721
1692 4E          MOV  C,M  } PUT ADDR OF 1ST CHAR OF ENTIRE FORMAT STRING IN HL          N    3722
1693 23          INX  H                                                                  #    3723
1694 66          MOV  H,M                                                                     3724
1695 69          MOV  L,C  }                                                                  3725
1696 16 00       MVI  D,H'00' CLEAR D FOR DAD INSTRUCTION                                     3726
1698 5F          MOV  E,A  PUT DIFFERENCE OF CHAR COUNTS IN E                            _    3727
1699 19          DAD  D COMPUTE ADDR OF NXT CHAR IN FORMAT STRING                            3728
169A 78          MOV  A,B  } PUT CHAR COUNT OF REMAINING FORMAT STRING IN B                  3729
169B B7          ORA  A    } & TEST IT                                                    7   3730
169C C2 9C 15    JNZ  X159C IF REMAINING STRING NOT USED-UP YET, CONTINUE WITH NXT FIELD OBTAINED FROM 3731 PRESENT POSITION IN STRING
169F C3 B1 16    JMP  X16B1 SKIP 2 INSTRUCTIONS                                         C1   3732
16A2 CD F4 16    CALL X16F4 PRINT LEADING '+' IF D≠Ø ('+' NOT A FORMAT CHAR IN THIS CASE) M   3733
16A5 DF          RST  3 PRINT CHAR IN ACC                                               _    3734
16A6 E1          POP  H PUT PTR TO INSTRUCTION IN HL                                         3735
16A7 F1          POP  PSW FETCH FLAG STATUS FROM STACK  ↓ IF IMMEDIATE INSTRUCTION FLAG ZERO = 1  3736 — GOES TO 16B6
16A8 C2 6B 15    JNZ  X156B IF LAST CHAR IN PRINT USING INSTRUCTION NOT REACHED YET, CONTINUE B WITH 3737 NXT ARG & START AT BEGINNING OF FORMAT
16AB DC 98 0A    CC   X0A98 IF CARRY-BIT SET (see 1685)                                 \    3738 STRING
16AE E3          XTHL PUT PTR TO IDENTITY GROUP OF FORMAT STRING IN HL, PUT PTR TO INSTRUCTION ON 3739 STACK
16AF CD 7D 13    CALL X137D ERADICATE FORMAT STRING FROM STRING AREA, REMOVE IDENTITY GROUP M FROM 3740 STRING FORMULA TABLE (IF THERE)
16B2 E1          POP  H PUT PTR TO INSTRUCTION IN HL                                         3741
16B3 C9          RET                                                                    I    3742
-------------------------------------------------------------------------------------------------------------------------------------
16B4 0E 01       X16B4 MVI  C,H'01' SET CHAR COUNT FOR LEFT$ (CALL 1305 AT 16DD) TO 1 — STRING 3743 FIELD COUNT = 1
16B6 3E F1       POP PSW MVI  A,H'F1' DUMMY REMOVE 1 ADDR FROM STACK (see 158Ø) - STRING FIELD 3744 COUNT IN C (ADDR OF 1ST '\'+1 NOT
16B8 05          DCR  B DECREMENT FORMAT STRING CHAR COUNT                                   3745                NEEDED SINCE 2ND '\' FOUND)
16B9 CD F4 16    CALL X16F4 PRINT LEADING '+' IF D≠Ø ('+' NOT A FORMAT CHAR IN THIS CASE) M  3746
16BC E1          POP  H PUT PTR TO INSTRUCTION IN HL (see 1571)                              3747
16BD F1          POP  PSW FETCH RESULT-OF-TEST (see 156F) STATUS FROM STACK                  3748
16BE CA 86 16    JZ   X1686 JMP IF PRINT USING INSTRUCTION IS IMMEDIATE (NOT IN A BASIC PROGRAM) J6  3749
16C1 C5          PUSH B SAVE FORMAT STRING CHAR COUNT (B) & STRING LENGTH (C) ON E STACK 3750
16C2 CD 52 0C    CALL X0C52 EVALUATE NXT ARG IN PRINT STMT TO VALUE IN HOLDING MR AREA 3751
16C5 CD 88 1C    CALL X1C88 TEST ARG VALUE - MUST BE STRING TYPE, OTHERWISE "TYPE M   3752 MISMATCH"
16C8 C1          POP  B  } PUT STRING FIELD COUNT IN C                                  A    3753
16C9 C5          PUSH B  } LEAVE FORMAT STRING CHAR COUNT ON STACK                      E    3754
16CA F5          PUSH H  PUT PTR TO INSTRUCTION ON STACK                                     3755
16CB 2A 12 04    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE (IDENTITY GROUP) * ON 3756 ARG STRING
16CE 41          MOV  B,C  PUT STRING FIELD COUNT (=1) IN B FOR CALL 1305            A    3757
16CF 0E 00       MVI  C,H'00' PUT 1ST CHAR POSITION OFFSET IN C (LEFT-MOST CHAR 3758 IS 1ST CHAR)
16D1 C5          PUSH B SAVE STRING FIELD COUNT FOR LEFT$ ON STACK                   E    3759
16D2 CD 05 13    CALL X1305 GENERATE STRING USING LEFT$-CHAR OF ARG STRING           MU   3760
16D5 CD 44 12    CALL X1244 PRINT THE NEWLY GENERATED STRING                         MD   3761
16D8 2A 12 04    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF NEW STRING       *    3762
16DB F1          POP  PSW PUT STRING FIELD COUNT IN ACC                                   3763
16DC 96          SUB  M   SUBTRACT CHAR COUNT BYTE OF NEW STRING - ALWAYS ≥ Ø             3764 RESULT (≠Ø IF ARG STRING NOT
16DD 47          MOV  B,A PUT DIFFERENCE COUNT IN B                                  G    3765 LONG ENOUGH TO FILL STRING FIELD)
16DE 3E 20       MVI  A,A' ' PUT A SPACE CHAR IN ACC                                 >    3766
16E0 DA DF 05    RST3 DCR B JC  X05DF DUMMY PRINT A SPACE - DECREMENT DIFFERENCE COUNT Z_  3767
16E3 C2 EC 16    JNZ  X16EC LOOP UNTIL SUFFICIENT SPACES HAVE BEEN PRINTED TO FILL ENTIRE 3768 STRING FIELD
16E6 C3 82 16    JMP  X1682 CONTINUE SCAN ARG STRING, IF NOT DONE CONTINUE WITH FORMAT STRING C  3769
-------------------------------------------------------------------------------------------------------------------------------------
16E9 F5          X16E9 PUSH PSW SAVE ACC ON STACK                                            3770
16EA 7A          MOV  A,D  } PUT BIT-FLAG BYTE IN ACC                                        3771 PRINT LEADING '+' IF D≠Ø
16EB B7          ORA  A    } & TEST IT                                                    7   3772 '+' NOT A FORMAT CHAR IN THAT
16EC 3E 2B       MVI  A,A'+' PUT '+' CHAR IN ACC                                        >+   3773 CASE
16EE C4 18 00    CNZ  X0018 IF BIT-FLAG BYTE ≠Ø, USE RST 3 TO PRINT '+' CHAR            D    3774
16F1 F1          POP  PSW RESTORE ACC FROM STACK                                             3775
16F2 C9          RET                                                                    I    3776
-------------------------------------------------------------------------------------------------------------------------------------
16F3 32 E9 03    X16F3 STA  X03E9 CLEAR ERROR BRANCH FLAG (see Ø4E8-Ø4ED)                2    3777
16F6 2A 07 03    LHLD X0307 LOAD HL WITH CURRENT LINE # FROM FLAG AREA                  *W   3778
16F9 B4          ORA  H  } TEST IF HL = FFFF                                            4    3779
```

```
1705  A5              ANA  L   } TEST IF HL = FFFF                          %   3780
1706  3C              INR  A   }                                            <   3781
1707  EB              XCHG     PUT CURRENT LINE # IN DE                        3782
1708  C8              RZ  RETURN IF LINE # IS FFFF - BASIC IS IN DIRECT COMMAND MODE - EDIT MODE NOT NEEDED H   3783
1709  C3 10 17        JMP  X1710 JMP INTO EDIT ROUTINE AUTOMATICLY IF SYNTAX ERROR OCCURRED WHILE RUNNING A   C   3784
170C  CD F3 08        CALL X08F3 ASSEMBLE NEXT GROUP OF CHAR INTO A 2 BYTE VALUE IN DE   M   3785 BASIC PROGRAM
170F  C0              RNZ  IF NON-SPACE CHAR AFTER LAST DIGIT IS NOT A ':' or NULL IGNORE   3786 (LINE #)
1710  E1        X1710 POP  H  REMOVE CNTL-C CHECK ADDRESS FROM STACK          3787 COMMAND
1711  CD 7B 05  X1711 CALL X057B SCAN STMT AREA TO FIND STMT WITH LINE # TO MATCH # M IN   3788 DE
1714  D2 71 09        JNC  X0971 RNC IF LINE # MATCH NOT FOUND - ERROR MESSAGE - "UNDEFINED   3789 STATEMENT"
1717  60              MOV  H,B } PUT ADDR OF FIRST BYTE OF STMT TO BE EDITED IN HL   3790
1718  69              MOV  L,C }                                            3791
1719  23              INX  H   } ADVANCE PTR TO POINT TO LINE #             #   3792
171A  23              INX  H   }                                            #   3793
171B  4E              MOV  C,M } FETCH LINE # FROM STMT                     N   3794
171C  23              INX  H   } PUT LINE # IN BC                           #   3795
171D  46              MOV  B,M } ADVANCE PTR TO POINT TO ASCII AREA OF STMT  F   3796
171E  23              INX  H   }                                            #   3797
171F  C5              PUSH B  PUT LINE # IN STACK                           E   3798
1720  CD E9 14        CALL X14E9 MOVE ASCII FROM STMT AREA TO LINE BUFR - EXPAND RESERVED M WORDS   3799
1723  E1        X1723 POP  H } PUT LINE # IN HL                                 3800
1724  F5              PUSH H } LEAVE LINE # ON STACK                          3801
1725  CD 73 21        CALL X2173 CONVERT LINE # TO DECIMAL ASCII FORM & PRINT M !   3802   HL - PTR TO LINE BUFR
1728  3E 20           MVI  A,A' ' } PRINT SPACE AFTER LINE #          >   3803   B - POSITION CTR OF EDITED LINE
172A  DF              RST  3   }                                      _   3804   C - CHAR CTR OF LINE BUFR (NULL NOT COUNTED)
172B  21 5A 03        LXI  H,X035A SET HL TO POINT TO FIRST BYTE IN LINE BUFFER! Z   3805   D - COMMAND CTR (REPEAT N TIMES)
172E  E5              PUSH H  PUT PTR ON STACK                                3806   E - SEARCH CHAR STORAGE
172F  0E FF           MVI  C,H'FF' INITIALIZE CHAR COUNT TO -1                3807
1731  0C        X1731 INR  C   INCR CHAR COUNT (not including null)           3808
1732  7E              MOV  A,M  FETCH CHAR FROM LINE BUFR                     3809
1733  E6 7F           ANI  H'7F' STRIP D7 FROM CHAR (MIGHT HAVE COME WITH LAST CHAR OF A   3810 RESERVED WORD)
1735  77              MOV  M,A  PUT CHAR BACK IN POSITION IN LINE BUFR        3811
1736  23              INX  H   ADVANCE LINE BUFR POINTER                  #   3812
1737  C2 31 17        JNZ  X1731 LOOP TIL ENTIRE BUFFER HAS BEEN SCANNED  B1  3813
173A  E1              POP  H  RESET HL TO POINT TO FIRST BYTE IN LINE BUFR    3814
173B  47              MOV  B,A CLEAR REG B                                G   3815
173C  16 00       X173C MVI  D,H'00' CLEAR REG D (DIGIT ACCUMULATOR - ASCII TO BINARY)   3816
173E  CC EC 06    X173E CALL X06EC FETCH 1 CHAR FROM INPUT CONSOLE        M   3817
1741  FE 3A           CPI  A':' } IF CHAR IS NOT NUMERIC JMP              :   3818
1743  D2 58 17        JNC  X1758 }                                        RX  3819
1746  FE 30           CPI  A'0'  }                                        0   3820
1748  DA 58 17        JC   X1758 }                                        ZX  3821
174B  D6 30           SUI  A'0' CONVERT ASCII TO BINARY NUMERIC VALUE     V0  3822
174D  5F              MOV  E,A  SAVE VALUE IN REG E                       -   3823   1 BYTE ASCII DIGIT STRING
174E  7A              MOV  A,D  PUT ACCUMULATED VALUE IN ACC                  3824   TO BINARY CONVERSION
174F  07              RLC  X2  } MULTIPLY VALUE BY 10                         3825
1750  07              RLC  X2  }                                              3826
1751  82              ADD  D+1 }                                              3827
1752  07              RLC  X2  }                                              3828
1753  83              ADD  E   ADD NEW VALUE                                  3829
1754  57              MOV  D,A STORE NEW ACCUMULATED SUM IN D              H  3830
1755  C3 3E 17        JMP  X173E                                           C>  3831
1758  E5        X1758 PUSH H  } PUT ADDR 173C ON STACK FOR IMPLIED JMP        3832
1759  21 3C 17        LXI  H,X173C }                                      !<  3833
175C  E3              XTHL                                                    3834
175D  15              DCR  D  } TEST DIGIT ACCUMULATOR                        3835
175E  14              INR  D  }                                               3836
175F  C2 63 17        JNZ  X1763 } IF DIGIT ACC IS ZERO, NO DIGITS WERE TYPED, SO SET B   3837 D TO 1
1762  14              INR  D  }                                               3838
1763  FE 20       X1763 CPI  A' ' } IF CHAR IS '␣' JMP                        3839
```

EDIT — AFTER 057B

LINE TO BE EDITED → B,C

CHN ADDR
LINE #
Ø
CHN ADDR
LINE #
Ø
CHN ADDR
LINE #

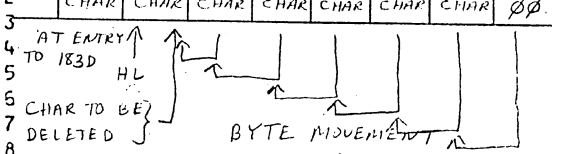
```
1765 CA 4F 17          JZ   X17AF  } IF CHAR IS 'U'
1768 FE 51             CPI  A'Q'  } IF CHAR IS 'Q'
176A CA F8 07          JZ   X07F8 }
176D FE 4C             CPI  A'L'  } IF CHAR IS 'L'
176F CA E9 17          JZ   X17E9 }
1772 FE 53             CPI  A'S'  } IF CHAR IS 'S'
1774 CA C0 17          JZ   X17C0 }
1777 FE 49             CPI  A'I'  } IF CHAR IS 'I'
1779 CA 21 18          JZ   X1821 }
177C FE 44             CPI  A'D'  } IF CHAR IS 'D'
177E CA F3 17          JZ   X17F3 }
1781 FE 0D             CPI  H'0D' } IF CHAR IS CR
1783 CA 7D 18          JZ   X187D }
1786 FE 43             CPI  A'C'  } IF CHAR IS 'C'
1788 CA 0A 18          JZ   X180A }
178B FE 45             CPI  A'E'  } IF CHAR IS 'E'
178D CA 80 18          JZ   X1880 }
1790 FE 58             CPI  A'X'  } IF CHAR IS 'X'
1792 CA 1C 18          JZ   X181C }
1795 FE 4B             CPI  A'K'  } IF CHAR IS 'K'
1797 CA BA 17          JZ   X17BA }
179A FE 48             CPI  A'H'  } IF CHAR IS 'H'
179C CA 19 18          JZ   X1819 }
179F FE 7F             CPI  H'7F' } IF CHAR IS RUBOUT
17A1 CA 71 18          JZ   X1871 }
17A4 FE 41             CPI  A'A'  } IF CHAR IS 'A' DO NOT RETURN RESTORE ORIGINAL LINE
17A6 C0                RNZ  IF NOT 1 OF ABOVE COMMANDS, IGNORE CHAR TYPED. FETCH ANOTHER
17A7 C1                POP  B   REMOVE 173C ADDR FROM STACK  THIS REMOVES UNWANTED ENTRY AFROM
17A8 D1                POP  D   REMOVE LINE # FROM STACK - THIS IS USED IN 057B          STACK
17A9 CD 98 0A          CALL X0A98 PRINT CRLF + NULLS
17AC C3 11 17          JMP  X1711 - LOOP BACK + START EDIT AGAIN

17AF 7E          X17AF MOV  A,M FETCH CHAR FROM LINE BUFR
17B0 B7                ORA  A   TEST CHAR
17B1 C8                RZ      IF NULL, END OF LINE  IMPLIED JMP TO 173C - FETCH ANOTHER
17B2 04                INR  B   INCREMENT CHAR CTR                                  COMMAND
17B3 CF                RST  3   PRINT CHAR FROM LINE BUFR ON OUTPUT CONSOLE
17B4 23                INX  H   INCREMENT LINE BUFR PTR
17B5 15                DCR  D   DECREMENT COMMAND CTR (ARG IN FRONT OF COMMAND, DEFAULT 1)
17B6 C2 AF 17          JNZ  X17AF IF COMMAND ARG IS NOT ZERO, REPEAT FOR ANOTHER CHAR
17B9 C9                RET      IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR

17BA E5          X17BA PUSH H     } PUT 1806 ON STACK FOR IMPLIED JMP          STACK
17BB 21 06 18          LXI  H,X1806 } HL STILL PTS TO LINE BUFR              173C
17BE E3                XTHL                                                  1806
17BF 37                STC      SET CARRY TO INDICATE 'K' COMMAND WHILE PASSING THRU 'S'

17C0 F5          X17C0 PUSH PSW   SAVE EDIT COMMAND CHAR ON STACK
17C1 CD EC 06          CALL X06EC FETCH SEARCH CHAR FROM INPUT CONSOLE
17C4 5F                MOV  E,A   PUT SEARCH CHAR IN REG E
17C5 F1                POP  PSW   RESTORE EDIT COMMAND CHAR TO ACC
17C6 35                DCR  M   } TEST CHAR IN LINE BUFR - DOES NOT AFFECT CARRY
17C7 34                INR  M   }
17C8 C8                RZ     IF CHAR IS NULL, END OF LINE, NTH SEARCH CHAR NOT FOUND YET
17C9 F5                PUSH PSW PUT EDIT COMMAND CHAR ON STACK WHILE ECHOING CHAR FROM LINE BUFR)
17CA DC 06 18          CC   X1806 PRINT '/' IF CARRY IS SET (CARRY IS SET BY 'K' COMMAND ABOVE)
17CD 7E                MOV  A,M FETCH CHAR FROM LINE BUFR
17CE CF          X17CE RST  3   PRINT CHAR ON OUTPUT CONSOLE
17CF F1                POP  PSW } RESTORE STATUS FLAGS.
17D0 F5                PUSH PSW } LEAVE EDIT COMMAND CHAR ON STACK
17D1 D2 C8 17          JNC  X17C8 CARRY NOT SET IF 'S' COMMAND
17D4 CC 30 18          CALL X1830 DELETE CHAR POINTED TO BY HL, ADJUST LINE BUFR + CHAR CTR (REG C)
17D7 C2 23 04    INX H INR B JNZ  X0423 DUMMY  ADVANCE LINE BUFFER PTR, INCREMENT EDITED LINE CHAR
                             FOR                                  FOR                  CTR
                            'K' COMMAND                          'S' COMMAND
```

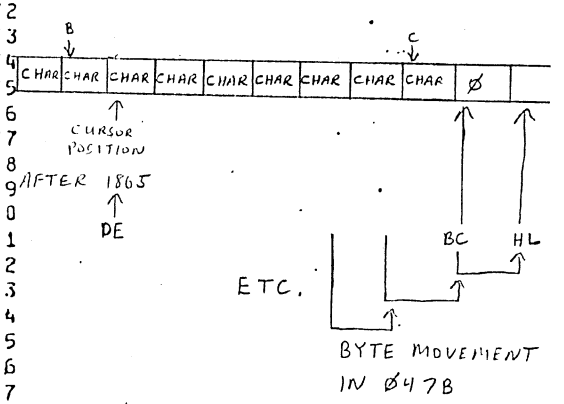
```
J/  3840  SKIP OVER A CHAR
Q   3841  RESTORE LINE TO ORIGINAL FORM, IGNORE EDIT CHANGES
J   3842
L   3843 + RE-ENTER EDIT MODE
J   3844
S   3845  SEARCH
J@  3846
I   3847  INSERT CHAR'S
J!  3848
D   3849  DELETE CHAR'S
J   3850
    3851  END OF EDIT, TYPE REST OF LINE
J   3852
C   3853  CHANGE CHAR'S IN LINE
J   3854
E   3855  END OF EDIT, DO NOT TYPE REST OF LINE
J   3856
X   3857  INSERT AT END OF LINE
J   3858
K   3859  SEARCH, DELETE CHAR'S PASSED OVER WHILE SEARCHING
J:  3860
H   3861  INSERT, DELETE REST OF LINE NOT YET TYPED
J   3862
    3863  DELETE A CHAR (DURING INSERT)
J   3864  RE-ENTER EDIT MODE
A + 3865  CHAR FROM INPUT CONSOLE
@   3866  FOR EDIT COMMAND
    3867  'A' COMMAND - RESTORE ORIGINAL LINE +
C   3868        RE-ENTER EDIT MODE
M   3869
C   3870

    3871  'U' COMMAND - SKIP OVER A CHAR
7   3872
HEDIT 3873
COMMAND 3874
-   3875
#   3876
B/  3877
I   3878
    3879

    3880  'K' COMMAND - SEARCH, DELETE CHAR'S
!   3881        PASSED OVER WHILE SEARCHING
    3882
7   3883

    3884  'S' COMMAND - SEARCH
M   3885
-   3886
    3887
5   3888
4   3889
H   3890
    3891
A   3892
    3893
-   3894
    3895
    3896
RX  3897
M=  3898
B   3899
```

```
170A 7E          MOV  A,M FETCH CHAR FROM LINE BUFFER ('S' COMMAND - NXT CHAR : 'K'-NXT CHAR
1709 B7          ORA  A   TEST CHAR
170C CA E7 17    JZ   X17E7 JMP IF CHAR IS NULL - END OF LINE, NTH CHAR NOT FOUND
170F B9          CMP  E   COMPARE CHAR FROM LINE BUFFER WITH SEARCH CHAR
1710 C2 CE 17    JNZ  X17CE IF NO MATCH, PRINT CHAR & TRY NEXT CHAR
1713 15          DCR  D   DECREMENT COMMAND CTR
1714 C2 CE 17    JNZ  X17CE IF NOT ZERO, NTH OCCURENCE OF SEARCH CHAR NOT FOUND
17E7 F1       X17E7 POP  PSW REMOVE EDIT COMMAND CHAR FROM STACK (SEARCH IS FINISHED OR
17E8 C3          RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR - CURSOR

17E9 CD E0 14 X17E9 CALL X14E0 PRINT REST OF LINE STARTING WITH CHAR POINTED TO BY HL
17EC CD 98 0A    CALL X0A98 PRINT CRLF & NULLS
17EF C1          POP  B     REMOVE 173C ADDR FROM STACK
17F0 C3 23 17    JMP  X1723 START EDIT ON NEW LINE - TYPE LINE #, ETC.

17F3 7E       X17F3 MOV  A,M  FETCH CHAR FROM LINE BUFR
17F4 B7          ORA  A     TEST CHAR
17F5 C8          RZ         RETURN IF CHAR IS NULL (CANNOT DELETE N CHAR'S IF NXT
17F6 3E 5C       MVI  A,A'\' } PRINT '\' ON OUTPUT CONSOLE
17F8 DF          RST  3     }
17F9 7E       X17F9 MOV  A,M  FETCH CHAR FROM BUFR
17FA B7          ORA  A     TEST CHAR
17FB CA 06 18    JZ   X1806 IF CHAR IS A NULL, COMMAND IS 'FINISHED
17FE DF          RST  3     PRINT CHAR ON OUTPUT CONSOLE
17FF CD 3D 18    CALL X183D DELETE CHAR POINTED TO BY HL, ADJUST LINE BUFR & CHAR CTR (REG
1802 15          DCR  D     DECREMENT COMMAND CTR
1803 C2 F9 17    JNZ  X17F9 IF COMMAND CTR IS NOT Ø, REPEAT FOR NXT CHAR
1806 3E 5C    X1806 MVI  A,A'\' } PRINT '/' ON OUTPUT CONSOLE
1808 DF          RST  3     }
1809 C9          RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR

180A 7E       X180A MOV  A,M  FETCH CHAR FROM LINE BUFR
180B B7          ORA  A     TEST CHAR
180C C8          RZ         JMP TO 173C IF NULL
180D CC EC 06    CALL X06EC FETCH CHAR TO PUT IN LINE BUFR AT CURSOR POSITION
1810 DF          RST  3     ECHO CHAR ON OUTPUT CONSOLE
1811 77          MOV  M,A   STORE CHAR AT CURSOR POSITION IN LINE BUFR
1812 23          INX  H     ADVANCE CURSOR POSITION
1813 04          INR  B     INCREMENT EDITED-LINE CHAR CTR
1814 15          DCR  D     DECREMENT COMMAND CTR
1815 C2 0A 18    JNZ  X180A LOOP TIL COMMAND CTR IS Ø
1818 C9          RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR

1819 36 00    X1819 MVI  M,H'00' PUT A NULL AT CURSOR POSITION - DELETES REST OF LINE
181B 48          MOV  C,B   PUT EDITED-LINE CHAR CTR IN LINE BUFR CHAR CTR

181C 16 FF    X181C MVI  D,H'FF' SET COMMAND CTR TO FF (MAXIMUM #)
181E CD AF 17    CALL X17AF SKIP OVER A CHAR ('␣' COMMAND) FF TIMES; ENTER INSERT

1821 CD EC 06 X1821 CALL X06EC FETCH A CHAR FROM INPUT CONSOLE
1824 FE 00       CPI  H'00' } IF CHAR IS CR - END OF EDIT; TYPE REST OF LINE
1826 CA 70 18    JZ   X1870 }
1829 FE 1B       CPI  H'1B' } IF CHAR IS 'ESCAPE' - IMPLIED JMP TO 173C, FETCH ANOTHER EDIT COMMAND
182B C8          RZ         }
182C FE 5F       CPI  A'_'  } JMP IF CHAR IS NOT A '←'
182E C2 4C 18    JNZ  X184C } IF CHAR IS A '←', DELETE CHAR
1831 05          DCR  B     } TEST EDITED LINE CHAR CTR  } TO LEFT OF CURSOR
1832 04          INR  B     }
1833 CA 54 18    JZ   X1854 IF CTR IS ZERO, NO CHAR'S TO LEFT OF CURSOR TO DELE
1836 DF          RST  3     ECHO '←' TO INDICATE A DELETED CHAR
1837 2B          DCX  H     BACK-UP CURSOR POSITION
1838 05          DCR  B     DECREMENT EDITED-LINE CHAR CTR
1839 11 21 18    LXI  D,X1821 } PUT ADDR 1821 ON STACK FOR IMPLIED JMP
183C D5          PUSH D       } (LOOP WITHIN INSERT COMMAND UNTIL '↵' or 'ESCAPE')
183D E5       X1830 PUSH H  SAVE CURSOR POSITION (PRESENT POSITION IN LINE BUFR)
183E 0D          DCR  C     DECREMENT CHAR COUNT BY 1
```

```
3900 BECAUSE BUFR WAS ADJUSTED DOWNWARD
3901
3902 'K' COMMAND WILL PRINT '/'
3903 THEN JMP TO 173C
3904
3905
3906
3907 END OF LINE REACHED)
3908 PTS TO NTH SEARCH CHAR OR NULL
3909 'L' COMMAND - TYPE REMAINDER OF LINE &
3910 RE-ENTER EDIT MODE
3911
3912
3913
3914 'D' COMMAND - DELETE CHAR'S
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928 'C' COMMAND - CHANGE CHAR'S IN LINE
3929
3930 CHANGE A CHAR AT CURSOR POSITION
3931 TO A CHAR TYPED AFTER 'C'
3932
3933
3934
3935
3936
3937
3938
3939 'H' COMMAND - INSERT MODE, DELETE REST OF
3940 LINE NOT YET TYPED
3941 'X' COMMAND - INSERT MODE AT END OF LINE
3942
3943 'I' COMMAND - INSERT CHAR'S
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
```

```
183F 7E          >X183F  MOV  A,M FETCH CHAR FROM LINE BUFFER                    3960
1840 B7                  ORA  A   TEST CHAR                                     3961
1841 CA 1C 1A      <--   JZ   X1A1C = POP H:RET - RESTORE PTR TO LINE BUFR       3962
1844 23                  INX  H  } FETCH NEXT CHAR FROM LINE BUFR               3963
1845 7E                  MOV  A,M}                                              3964
1846 2B                  DCX  H   BACK-UP PTR 1 POSITION                        3965
1847 77                  MOV  M,A STORE CHAR IN NEW POSITION IN BUFR            3966
1848 23                  INX  H   ADVANCE PTR                                   3967
1849 C3 3F 18            JMP  X183F LOOP TIL NULL FOUND                         3968
184C F5          >X184C  PUSH PSW SAVE CHAR TO INSERT ON STACK                  3969
184D 79                  MOV  A,C  } TEST CHAR CTR OF LINE BUFR                 3970
184E FE 48               CPI  A'H' } MUST BE LESS THAN 72 FOR NEW CHAR TO BE INSERTED  3971
1850 DA 5A 18            JC   X185A JMP IF ENOUGH ROOM IN LINE BUFR FOR AN ADDITIONAL CHAR  3972
1853 F1                  POP  PSW REMOVE CHAR FROM STACK - RING BELL + IGNORE FURTHER INSERTED CHARS  3973
1854 3E 07       >X1854  MVI  A,H'07' } PRINT CNTL-G ON OUTPUT CONSOLE (RING BELL)  3974
1856 DF                  RST  3       }                                         3975
1857 C3 21 18            JMP  X1821  JMP BACK TO START OF INSERT COMMAND        3976
185A 90          >X185A  SUB  B  SUBTRACT CURSOR POSITION COUNT FROM LINE LENGTH  3977
185B 0C                  INR  C  INCREMENT CHAR CTR OF LINE BUFR                3978
185C 04                  INR  B  INCREMENT EDITED-LINE CHAR CTR                 3979
185D C5                  PUSH B  SAVE BOTH CTR'S ON STACK                       3980
185E EB                  XCHG    PUT PTR TO LINE BUFR (CURSOR POSITION) IN DE   3981
185F 6F                  MOV  L,A  } COMPUTE ADDR OF LAST CHAR IN BUFR (ADDR OF NULL)  3982
1860 26 00               MVI  H,H'00'}                                          3983
1862 19                  DAD  D                                                 3984
1863 44                  MOV  B,H  } PUT ADDR OF LAST CHAR IN BUFR IN BC        3985
1864 4D                  MOV  C,L  }                                            3986
1865 23                  INX  H   ADVANCE PTR TO POINT TO BYTE AFTER NULL IN LINE BUFR  3987
1866 CC 78 04            CALL X0478 MOVE CHAR IN LINE BUFR TO MAKE ROOM FOR NEW CHAR  3988
1869 C1                  POP  B  RESTORE CTR VALUES TO B & C                    3989
186A F1                  POP  PSW PUT CHAR TO INSERT BACK INTO ACC              3990
186B DF                  RST  3  ECHO CHAR ON OUTPUT CONSOLE                    3991
186C 77                  MOV  M,A STORE CHAR AT CURSOR POSITION IN LINE BUFR    3992
186D 23                  INX  H  ADVANCE CURSOR POSITION                        3993
186E C3 21 18            JMP  X1821 LOOP WITHIN INSERT UNTIL 'D' OR 'ESCAPE'    3994
1871 78          >X1871  MOV  A,B  } TEST EDITED-LINE CHAR COUNT                3995 'RUBOUT' COMMAND - DELETE A CHAR DURING INSERT
1872 B7                  ORA  A    }                                            3996
1873 C8                  RZ     IF ZERO - CANNOT DELETE ANY CHAR'S TIL CURSOR IS MOVED  3997
1874 05                  DCR  B  DECREMENT EDITED LINE CHAR COUNT               3998
1875 2B                  DCX  H  BACK-UP PTR 1 CHAR                             3999
1876 7E                  MOV  A,M } FETCH CHAR FROM BUFR                        4000
1877 DF                  RST  3   } ECHO CHAR ON OUTPUT CONSOLE                 4001
1878 15                  DCR  D   DECREMENT COMMAND CTR                         4002
1879 C2 71 18            JNZ  X1871 IF NOT ZERO, REPEAT FOR ANOTHER CHAR        4003
187C C9                  RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR  4004
187D CD E0 14    X187D   CALL X14E0 PRINT REST OF LINE STARTING WITH CHAR POINTED TO BY HL  4005 'CR' COMMAND - END OF EDIT, TYPE REST OF LINE
1880 CC 98 0A    X1880   CALL X0A98 PRINT CRLF + NULLS                          4006 'E' COMMAND - END OF EDIT, DO NOT TYPE REST OF LINE
1883 C1                  POP  B REMOVE ADDR 173C FROM STACK (IMPLIED JMP NO LONGER NEEDED)  4007
1884 D1                  POP  D PUT LINE # OF EDITED STMT IN DE (FOR ROUTINE TO STORE EDITED LINE)  4008
1885 37                  STC    } FORCE CARRY SET & PUT CARRY STATUS ON STACK   4009 :
1886 F5                  PUSH PSW} (FOOLS COMMAND MODE MODE PROCESSOR, STRING IN LINE BUFR NOT TO BE TREATED AS  4010 A DIRECT COMMAND)
1887 21 5A 03            LXI  H,X035A LOAD HL WITH ADDR OF START OF LINE BUFR   4011
188A C3 D1 1B            JMP  X1BD1 JMP TO COMMAND MODE PROCESSING ROUTINE (MOVE EDITED LINE FROM LINE BUFR  4012 TO STMT AREA - SAME AS NEW LINE FROM KYBD)
188D 21 49 29    X1890   LXI  H,X2949 LOAD HL WITH ADDR OF SNG PREC CONSTANT = .5  4013 ADD .5 TO # IN HOLDING AREA
1890 CD 54 19    X1890   CALL X1954 PUT SNG-PREC CONSTANT IN BCDE  B = EXP BYTE (FETCH FROM MEMORY)  4014
1893 C3 9C 18            JMP  X189C ADD CONSTANT = .5 TO # IN HOLDING AREA (or other constant)  4015
1896 CD 54 19    X1896   CALL X1954 PUT SNG-PREC CONSTANT IN BCDE  (FETCH FROM MEMORY)  4016
1899 CD 13 1D    X1899   CALL X1D13 NEGATE FLOATING-PT NUMBER IN HOLDING AREA  4017 SNG-PREC FLOATING-PT SUBTRACT
189C CD 13 1D    X189C   CALL X1D13 NEGATE FLOATING-PT NUMBER IN HOLDING AREA  4018 SNG-PREC FLOATING-PT ADDITION
189F 78          X189F   MOV  A,B  } TEST EXPONENT BYTE OF # IN BCDE            4019 (# IN BCDE) + (# IN 1D12...) ADD
                                                                                    (# IN BCDE) - (# IN 1D12...) SUB
```





```
189E  C8               RZ    ADD OR SUB DONE IF # IS Ø, RESULT ALREADY IN HOLDING AREA   H     (<- DONE, # IS ZERO)
189F  3A 15 04         LDA   X0415} TEST EXPONENT BYTE OF # IN HOLDING AREA   :
18A2  B7               ORA   A    }   7
18A3  CA 46 18         JZ    X1846 ADD OR SUB DONE IF # IS Ø, STORE (BCDE) RESULT IN HOLDING AREA   JF   (<- DONE, OTHER # IS ZERO)
18A6  90               SUB   B SUBTRACT EXPONENT BYTES TO TEST RELATIVE SIZE OF #'S
18A7  D2 B6 18         JNC   X18B6 JMP IF (# IN HOLDING AREA) ≥ (# IN BCDE) [EXPONENTS]   R6
18AA  2F               CMA   } FORM 2'S COMPLEMENT OF DIFFERENCE OF EXPONENTS   /
18AB  3C               INR   A } (POSITIVE # IN ACC AFTERWARDS)   <
18AC  EB               XCHG  SAVE CONTENTS OF DE IN HL
18AD  CD 36 1B         CALL  X1B36 PUT FLOATING-PT # IN HOLDING AREA ON STACK   M6
18B0  EB               XCHG  RESTORE CONTENTS OF DE FROM HL
18B1  CD 46 1B         CALL  X1B46 PUT FLOATING-PT # IN BCDE INTO HOLDING AREA   MF
18B4  C1               POP   B} FETCH FLOATING-PT # FROM STACK INTO BCDE, B = EXP BYTE   A
18B5  D1               POP   D}   Q
          (SWAP FLOATING-PT # IN BCDE WITH # IN HOLDING AREA
           RESULT: # IN HOLDING AREA IS THE LARGER #, ACC HAS POSITIVE #
           EQUAL TO DIFFERENCE IN EXPONENTS)
18B6  FE 19     X18B6  CPI   H'19'} TEST IF EXPONENT DIFFERENCE IS 25. OR GREATER.
18B8  D0               RNC   } IF IT IS, # IN BCDE TOO SMALL TO AFFECT ANSWER (NOTE: 23 BITS OF VALUE + 1 BIT IMPLIED = 24)   (<- DONE, # TOO SMALL)
18B9  F5               PUSH  PSW SAVE EXPONENT DIFFERENCE ON STACK
18BA  CD 73 1B         CALL  X1B73 COMPUTE XNOR OF SIGN BITS IN ACC, RESTORE LEADING 1 IN MANTISSA'S OF BOTH #'S   M
18BD  67               MOV   H,A SAVE XNOR PRODUCT IN H
18BE  F1               POP   PSW PUT EXPONENT DIFFERENCE INTO ACC FROM STACK
18BF  CD 63 19         CALL  X1963 SHIFT SMALLER MAGNITUDE # IN CDE RIGHT (# IN ACC) PLACES, MSB OF   REG B IS LOST BIT OF SMALLER #
18C2  B4               ORA   H ACC IS Ø, PUT H IN ACC, SET FLAGS, H HAS $\overline{S_m} \oplus S_R$ IN THE MSB OF ACC   4
18C3  21 12 04         LXI   H,X0412 LOAD HL WITH ADDR OF LSB OF # IN HOLDING AREA   !
18C6  F2 DC 18         JP    X18DC JMP IF THE SIGN BITS ARE OPPOSITE   \
18C9  CD 43 19         CALL  X1943 ADD ADJUSTED SMALLER # IN CDE TO # IN Ø412-Ø414, RESULT IN CDE   MC
18CC  D2 22 19         JNC   X1922 IF NO CARRY, ADDITION DID NOT PRODUCE OVERFLOW, PERFORM ROUNDOFF + NORMALIZE RESULT   R"
18CF  23               INX   H ADVANCE HL TO POINT TO EXPONENT BYTE Ø415   #   (<- ALMOST DONE)
18D0  34               INR   M ADD 1 TO EXPONENT OF # IN HOLDING AREA (EXP OF RESULT)   4
18D1  CA 3E 19         JZ    X193E IF EXPONENT IS NOW ZERO, PRINT ERROR MESSAGE "OVERFLOW"   J>   (<- ERROR)
18D4  2E 01            MVI   L,H'01' SET SHIFT-RIGHT CTR TO 1 (DIVIDE MAGNITUDE BY 2 SINCE ADDING 1 TO EXPONENT SAME AS MULTIPLY BY 2)
18D6  CD 79 19         CALL  X1979 SHIFT MAGNITUDE OF RESULT IN CDE RIGHT 1 PLACE, OVERFLOW BIT IN CARRY SHIFTED INTO HI-ORDER BIT OF REG C   M
18D9  C3 22 19         JMP   X1922 PERFORM ROUNDOFF + NORMALIZE RESULT.   C"
------------------------------------------------------------------------
18DC  AF        X18DC  XRA   A CLEAR ACC   /
18DD  90               SUB   B  }
18DE  47               MOV   B,A}   G
18DF  7E               MOV   A,M}
18E0  9B               SBB   E  }
18E1  5F               MOV   E,A}
18E2  23               INX   H  }   -
18E3  7E               MOV   A,M}   #
18E4  9A               SBB   D  }
18E5  57               MOV   D,A}
18E6  23               INX   H  }   W
18E7  7E               MOV   A,M}   #
18E8  99               SBB   C  }
18E9  4F               MOV   C,A}   O

          COMPUTE DIFFERENCE OF THE 2 MANTISSA'S
            (Ø414)   (Ø413)   (Ø412)    ØØ
             - C      - D      - E      - B   <- LOST-BIT BYTE
             ----     ----     ----     ----
               C        D        E        B

18EA  DC 4F 19  X18EA  CC    X194F IF CARRY SET (OVERFLOW) (OR MINUS #), CHANGE SIGN OF RESULT (STORED AT Ø416 BY \ X1B73), + COMPUTE Ø - (CDEB)
------------------------------------------------------------------------
18ED  68        X18ED  MOV   L,B} CHANGE REGISTERS HOLDING # FROM CDEB TO CDHL (SPEEDS UP BIT SHIFT ROUTINE)
18EE  63               MOV   H,E}
18EF  AF               XRA   A CLEAR ACC (THIS LOOP ALLOWS 4 1-BYTE (8 BIT) LEFT SHIFTS AS PART OF NORMALIZING
18F0  47        X18F0  MOV   B,A SAVE SHIFT CTR IN B (COUNTS # OF BIT SHIFTS - USED TO ADJUST EXPONENT BYTE)   G
18F1  79               MOV   A,C} PUT HI-ORDER BYTE OF # TO BE NORMALIZED IN ACC + TEST IT
18F2  B7               ORA   A  }   7
18F3  C2 0F 19         JNZ   X190F IF HI-ORDER BYTE IS NOT ZERO, THEN FULL-BYTE SHIFT NOT APPLICABLE, JMP TO SINGLE BIT SHIFT ROUTINE   B
18F6  4A               MOV   C,D}   # IN  C D H L ØØ(ACC)   J
18F7  54               MOV   D,H}                <-<-<-<-      T
18F8  65               MOV   H,L}
18F9  6F               MOV   L,A}
18FA  78               MOV   A,B PUT SHIFT CTR IN ACC FROM B
18FB  D6 08            SUI   H'08' SUBTRACT 8 (EACH BYTE SHIFT THE SAME AS 8 BIT SHIFTS, MUST SUBTRACT
                             1 FROM EXPONENT FOR EACH SHIFT)

          NORMALIZING A # IN CDEB BY LEFT-SHIFTING
```

```
18FD  FE E0          CPI   H'E0' TEST IF 4 BYTE-SHIFTS HAVE BEEN DONE YET                          4080
18FF  C2 F0 18       JNZ   X18F0 JMP TO 18F0 IF MORE SHIFTS TO DO, OTHERWISE ALL 4 BYTES WERE FOUND TO BE 4081 Ø, THEREFORE RESULT IS Ø
1902  AF          X1902  XRA   A     CLEAR ACC                                                    4082
1903  32 15 04    X1903  STA   X0415 FORCE FLOATING-PT VALUE IN HOLDING AREA TO ZERO (EXP BYTE = Ø)  4083
1906  C9                 RET                                                                      4084
                                                                                                  4085
1907  05          X1907  DCR   B DECREMENT SHIFT CTR (EACH SHIFT SAME AS MULTIPLY BY 2, SUBTRACT 1 FROM EXPONENT TO COMPENSATE)
1908  29                 DAD   H DO A QUICK LEFT SHIFT (1 BIT LEFT) ON THE 2 LO BYTES IN HL (# IS IN CDHL)  4086
1909  7A                 MOV   A,D  } SHIFT BYTE IN D LEFT 1 BIT                                  4087
190A  17                 RAL        }                                                             4088
190B  57                 MOV   D,A  }                                                             4089
190C  79                 MOV   A,C  } SHIFT BYTE IN C LEFT 1 BIT                                  4090
190D  8F                 ADC   A    }                                                             4091
190E  4F                 MOV   C,A  }                                                             4092
190F  F2 07 19      X190F  JP    X1907 IF MSB OF # IS NOT A 1, THEN JMP TO 1907 TO DO A 1 BIT LEFT SHIFT  4093
1912  78                 MOV   A,B PUT SHIFT CTR IN ACC FROM B                                    4094
1913  5C                 MOV   E,H  } CHANGE REGISTERS HOLDING # FROM CDHL TO CDEB (BACK TO STANDARD FORMAT)  4095
1914  45                 MOV   B,L  }                                                             4096
1915  B7                 ORA   A TEST SHIFT CTR                                                   4097
1916  CA 22 19           JZ    X1922 IF SHIFT CTR IS ZERO, PERFORM ROUNDOFF, STORE RESULT IN HOLDING AREA  4098
1919  21 15 04           LXI   H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE                          4099
191C  86                 ADD   M  } COMPUTE EXPONENT = EXPONENT - (# OF SHIFTS TO NORMALIZE).     4100
191D  77                 MOV   M,A}                                                               4101
191E  D2 02 19           JNC   X1902 IF NO CARRY, NORMALIZED # IS SO SMALL THAT IT IS SAME AS Ø   4102
1921  C8            DONE RZ    IF EXPONENT BYTE IS ZERO, NORMALIZED # SAME AS Ø, ADD IS DONE      4103
                                                                                                  4104
1922  78          X1922  MOV   A,B PUT BYTE CONTAINING BIT LOST BY SHIFTING INTO ACC              4105 RECOMBINE SIGN-OF-RESULT WITH MSB
1923  21 15 04    X1923  LXI   H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE OF RESULT IN HOLDING AREA  4106 OF MANTISSA OF RESULT
1926  B7                 ORA   A TEST LOST BIT (MSB OF ACC)                                       4107 TO THE # IN CDE
1927  FC 34 19           CM    X1934 IF BIT SHIFTED OUT OF SMALLER MANTISSA WAS A 1, ROUND UP BY ADDING 1  4108 at 1931
192A  46                 MOV   B,M PUT EXPONENT BYTE OF RESULT (LARGER # EXPONENT) IN B (FOR 1B46)  4109
192B  23                 INX   H ADVANCE PTR TO LOC Ø4/6 (see 1B73 ROUTINE)                       4110
192C  7E                 MOV   A,M PUT BYTE WITH SIGN OF LARGER # (OR # IN HOLDING AREA) IN ACC   4111
192D  E6 80              ANI   H'80' ELIMINATE ALL BUT THE SIGN BIT                               4112 XNOR USED IN 1B73 SINCE LEADING 1 IN
192F  A9                 XRA   C COMBINE SIGN BIT WITH HI 7 BITS OF MANTISSA OF RESULT (NOTE: LEADING 1 LOST,  4113 192F INVERTS THE SIGN BIT
1930  4F                 MOV   C,A PUT BYTE BACK INTO C                                           4114
1931  C3 46 19           JMP   X1B46 STORE RESULT (IN BCDE) INTO HOLDING AREA                     4115 ADD 1 TO MANTISSA IN CDE
1934  1C          X1934  INR   E ADD 1 TO LO-ORDER BYTE                                           4116
1935  C0                 RNZ   IF NOT ZERO, DONE   IF ZERO, CARRY 1 INTO NEXT BYTE               4117 (24 BIT MANTISSA, EXPONENT
1936  14                 INR   D ADD 1 TO MID-BYTE                                                4118
1937  C0                 RNZ   IF NOT ZERO, DONE   IF ZERO, CARRY 1 INTO NEXT BYTE               4119 OF RESULT SAME AS EXPONENT
1938  0C                 INR   C ADD 1 TO HI-ORDER BYTE                                           4120 OF LARGER #, IF OVERFLOW,
1939  C0                 RNZ   IF NOT ZERO, DONE   IF ZERO, ADD 1 TO EXP, FORCE MANTISSA TO 80 00 00  4121
193A  0E 80              MVI   C,H'80' DIVIDE ENTIRE MANTISSA BY 2 (PUTS OVERFLOW BACK IN) MANTISSA  4122 ADD 1 TO EXPONENT, FORCE
193C  34                 INR   M ADJUST EXPONENT TO COMPENSATE                                    4123
193D  C0                 RNZ   IF NOT ZERO, DONE   IF ZERO, EXPONENT BYTE OVERFLOW ↓              4124 MANTISSA TO 80 00 00
193E  1E 06         X193E  MVI   E,H'06' } PRINT ERROR MESSAGE "OVERFLOW"                         4125
1940  C3 B7 04           JMP   X04B7 }                                                            4126
                                                                                                  4127 ADD A 3 BYTE # STORED IN
1943  7E          X1943  MOV   A,M  } ADD LO BYTE OF # TO E                                       4128 MEMORY TO C D E REG'S
1944  83                 ADD   E    }                                                             4129
1945  5F                 MOV   E,A  }                                                             4130
1946  23                 INX   H ADVANCE PTR TO #                                                 4131 NOTE: CARRY BIT IS USED
1947  7E                 MOV   A,M  }                                                             4132
1948  8A                 ADC   D    } ADD MED BYTE OF # TO D                                      4133
1949  57                 MOV   D,A  }                                                             4134
194A  23                 INX   H ADVANCE PTR TO #                                                 4135
194B  7E                 MOV   A,M  }                                                             4136
194C  89                 ADC   C    } ADD HI BYTE OF # TO C                                       4137
194D  4F                 MOV   C,A  }                                                             4138 CONVERT MANTISSA TO A POSITIVE #
194E  C9                 RET                                                                      4139 = TO MAGNITUDE OF RESULT
194F  21 16 04      X194F  LXI   H,X0416 LOAD HL WITH ADDR OF BYTE CONTAINING SIGN OF RESULT (see 1B73)!
1952  7E                 MOV   A,M  } COMPLEMENT SIGN BIT OF RESULT  → SIGN OF ARG IN HOLDING AREA
```

95

```
1953  2F                    CMA          } COMPLEMENT SIGN BIT
1954  77                    MOV  M,A     }   OF RESULT
1955  AF                    XRA  A       CLEAR ACC
1956  6F                    MOV  L,A     SAVE ØØ IN L
1957  90                    SUB  B
1958  47                    MOV  B,A     } COMPUTE  ØØ ØØ ØØ ØØ
1959  79                    MOV  A,L     }            -C -D -E -B
195A  98                    SBB  E       }
195B  5F                    MOV  E,A     }
195C  79                    MOV  A,L     }
195D  9A                    SBB  D       }
195E  57                    MOV  D,A     }
195F  79                    MOV  A,L     }
1960  99                    SBB  C       }
1961  4F                    MOV  C,A     }
1962  C9                    RET

1963  06 00          X1963  MVI  B,H'00' CLEAR REG B - ACCUMULATES BITS AS THEY ARE SHIFTED OUT OF # BEING RIGHT ADJUSTED
1965  D6 08          X1965, SUI  H'08'  TEST IF EXPONENT DIFFERENCE IS 7 OR LESS
1967  DA 72 19             JC   X1972 IF DIFFERENCE IS 7 OR LESS, DO A MULTIPLE REGISTER SHIFT-RIGHT TO ZFINISH
196A  43                    MOV  B,E   } IF DIFFERENCE IS 8 or larger, shuffle entire bytes to
196B  5A                    MOV  E,D   } perform equivalent shift right 8 times (much faster)
196C  51                    MOV  D,C   }
196D  0E 00                 MVI  C,H'00' } ← hi order byte forced to ØØ
196F  C3 65 19              JMP  X1965 REPEAT EXPONENT DIFFERENCE TEST TIL DIFFERENCE IS LESS THAN 8
1972  C6 09          X1972  ADI  H'09' ADJUST EXPONENT DIFFERENCE TO POSITIVE # (1 LARGER SINCE LOOP DECR'S FCTR FIRST)
1974  6F                    MOV  L,A   PUT # TO COUNT LOOPS IN REG L
1975  AF                X1975 XRA  A CLEAR CARRY BIT (FOR ROTATE RIGHT - 1979)
1976  2D                    DCR  L DECREMENT CTR (COUNTS # OF TIMES TO SHIFT # IN CDE RIGHT)
1977  C8                DONE RZ   IF CTR IS ZERO, ADJUSTMENT IS FINISHED
1978  79                    MOV  A,C PUT HI-BYTE IN ACC
1979  1F                X1979 RAR     SHIFT RIGHT, LO-BIT INTO CARRY
197A  4F                    MOV  C,A RESULT INTO C
197B  7A                    MOV  A,D PUT MID-BYTE IN ACC
197C  1F                    RAR      SHIFT RIGHT, BIT IN CARRY INTO HI-BIT, LO-BIT INTO CARRY
197D  57                    MOV  D,A RESULT INTO D
197E  7B                    MOV  A,E PUT LO-BYTE IN ACC
197F  1F                    RAR      SHIFT RIGHT, BIT IN CARRY INTO HI-BIT, LO-BIT INTO CARRY
1980  5F                    MOV  E,A RESULT INTO E
1981  78                    MOV  A,B PUT ACCUMULATED LOST-BITS IN ACC
1982  1F                    RAR      SHIFT RIGHT, ANOTHER LOST BIT INTO REG B
1983  47                    MOV  B,A RESULT INTO B
1984  C3 75 19              JMP  X1975 REPEAT LOOP TIL ENTIRE # ADJUSTED

1987  00          X1987     NOP  } SINGLE PRECISION CONSTANT
1988  00                    NOP  }   = 1
1989  00                    NOP  }
198A  81                    ADD  C

198B  03          X198B     INX  B CONSTANT TO INDICATE 3 SINGLE-PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE
198C  AA                    XRA  D     } SINGLE PRECISION CONSTANT
198D  56                    MOV  D,M   }   = .598979
198E  19                    DAD  D     }
198F  80                    ADD  B     }

1990  F1                    POP  PSW   } SINGLE PRECISION CONSTANT
1991  22 76 80              SHLD X8076 }   = .961471

1994  45                    MOV  B,L   } SINGLE PRECISION CONSTANT
1995  AA                    XRA  D     }   = 2.88539
1996  38                    DATA A'8'  }
1997  82                    ADD  D     }

1998  EF          X1998     RST  5  TEST SIGN OF ARG IN HOLDING AREA
1999  EA EE 08              JPE  X08EE IF ARG IS Ø OR NEGATIVE, PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"
199C  21 15 04              LXI  H,X0415 } LOAD ACC WITH EXPONENT BYTE OF ARG IN HOLDING AREA
```

Shift # in CDE Right n places

n in ACC

ADJUSTING #

C	D	E	B
Hi Byte	Mid Byte	Lo Byte	Ø Ø

before adjust — Bits Lost by Shift-Right

C	D	E	B
ØØ...	#		

MULTIPLE REG SHIFT RIGHT

NOTE: MAGNITUDE IS 24 BITS
SIGN BIT HAS BEEN EXTRACTED

LOG

```
                                    } LOAD ACC WITH EXPONENT BYTE OF ( ) ARG IN HOLDING AREA
199F  7E          MOV  A,M          }                                                    4200
19A0  01 35 80    LXI  B,X8035      } LOAD BCDE WITH SNG-PREC CONSTANT = √2/2      5    4201
19A3  11 F3 04    LXI  D,X04F3      }                                                    4202
19A6  90          SUB  B COMPUTE POWER OF 2 REQUIRED TO ADJUST ARG TO THE RANGE .5 ≤ ARG < 1   4203
19A7  F5          PUSH PSW SAVE ARG MULTIPLIER ON STACK                                4204
19A8  70          MOV  M,B FORCE ARG TO THE RANGE .5 ≤ ARG < 1 BY FORCING EXPONENT TO 80 (= 2⁻¹) 4205
19A9  D5          PUSH D  }                                                      U     4206
19AA  C5          PUSH B  } PUT CONSTANT = √2/2 ON STACK                         E     4207
19AB  CD 9C 18    CALL X189C COMPUTE ARG + √2/2                                  M     4208
19AE  C1          POP  B  } PUT CONSTANT = √2/2 IN BCDE FROM STACK               A     4209
19AF  D1          POP  D  .                                                      Q     4210
19B0  04          INR  B MULTIPLY CONSTANT BY 2 (CONSTANT NOW = √2)                    4211
19B1  CD 2E 1A    CALL X1A2E COMPUTE √2/(ARG + √2/2)                             M.    4212
19B4  21 87 19    LXI  H,X1987 LOAD HL WITH ADDR OF CONSTANT = 1                 !     4213
19B7  CD 96 18    CALL X1896 COMPUTE 1 - [√2/(ARG + √2/2)] =  (ARG - √2/2)/(ARG + √2/2)  M  4214
19BA  21 88 19    LXI  H,X1988 LOAD HL WITH ADDR OF POLYNOMIAL TABLE FOR LOG     !     4215
19BD  CD 58 26    CALL X2658 EVALUATE POLYNOMIAL                                 MX&   4216
19C0  01 80 80    LXI  B,X8080      } LOAD BCDE WITH SNG-PREC CONSTANT = -.5            4217
19C3  11 00 00    LXI  D,X0000      }                                                   4218
19C6  CD 9C 18    CALL X189C ADD CONSTANT TO POLYNOMIAL                          M     4219
19C9  F1          POP  PSW FETCH ARG MULTIPLIER FROM STACK                             4220
19CA  CD 4C 21    CALL X214C CONVERT ARG MULTIPLIER TO SNG-PREC, ADD TO POLYNOMIAL   ML!   4221
19CD  01 31 80    X19CD LXI  B,X8031  } LOAD BCDE WITH SNG-PREC CONSTANT = .693147 = LN 2  1  4222
19D0  11 18 72    LXI  D,X7218        } TO FINISH LOG, MULTIPLY POLYNOMIAL BY LN 2      4223
-------------------------------------------------------------------------------------------
19D3  EF          X19D3 RST  5 PERFORM FLOATING-PT SIGN TEST ON # IN HOLDING AREA       4224   SINGLE-PRECISION MULTIPLY
19D4  C8          RZ   IF # IS ZERO, THEN MULTIPLY IS DONE & RESULT IS ALREADY IN HOLDING AREA 4225   (# IN BCDE) × (# IN HOLDING AREA)
19D5  2E 00       MVI  L,H'00' SET L FOR MULTIPLY ROUTINE (SUM OF EXPONENTS)          4226
19D7  CD AB 1A    CALL X1AAB COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN  M+BITS 4227   MULTIPLICAND (1ST #)   MULTIPLIER (2ND #)
19DA  79          MOV  A,C  } STORE HI-8 BITS OF MULTIPLICAND IN ACI INSTRUCTION AT 1AB9  4228
19DB  32 0A 1A    STA  X1A0A}                                                    2     4229
19DE  E9          XCHG      } STORE LO-16 BITS OF MULTIPLICAND IN LXI INSTRUCTION AT 1AB4  4230
19DF  22 05 1A    SHLD X1A05}                                                    "     4231
19E2  01 00 00    LXI  B,X0000} USE CDEB TO ACCUMULATE MANTISSA OF PRODUCT              4232
19E5  50          MOV  D,B    } INITIALLY SET ACCUMULATED VALUE TO 00 00 00 00    P     4233
19E6  58          MOV  E,B    } (B IS LOST-BIT BYTE)                             X     4234
19E7  21 FD 18    LXI  H,X18FD} PUT ADDR OF (ROUTINE TO NORMALIZE # IN CDEB, ADJUST EXPONENT,  !  4235
19EA  E5          PUSH H      } THEN PUT RESULT IN HOLDING AREA) ON STACK - LAST THING TO DO AT END OF MULTIPLY  4236
19EB  21 F3 19    LXI  H,X19F3} PUT ADDR 19F3 ON STACK (TOTAL-LOOP 3 TIMES - 3 BYTES IN MULTIPLIER)  !  4237
19EE  E5          PUSH H      }                                                        4238
19EF  E5          PUSH H      }                                                        4239
19F0  21 12 04    LXI  H,X0412 INITIALLY SET HL TO POINT TO LO-ORDER BYTE OF THE MULTIPLIER IN HOLDING AREA  4240
19F3  7E          X19F3 MOV  A,M FETCH BYTE OF MULTIPLIER INTO ACC                   !     4241
19F4  23          INX  H  ADVANCE PTR TO NEXT BYTE IN MULTIPLIER                     #     4242
19F5  87          ORA  A  TEST MULTIPLIER BYTE                                       7     4243
19F6  CA 1E 1A    JZ   X1A1E IF MULTIPLIER BYTE IS 00, THEN NOTHING TO ACCUMULATE IN PRODUCT, SHIFT ACCUMULATED PRODUCT 8 BITS RIGHT  J  4244
19F9  E5          PUSH H SAVE PTR TO MULTIPLIER ON STACK                              4245
19FA  EB          XCHG PUT 2-LO ORDER BYTES OF ACCUMULATED PRODUCT IN HL (SPEEDS UP BIT SHIFT ROUTINE)  4246
19FB  1E 08       MVI  E,H'08' SET BIT-LOOP CTR TO 8 (8 BITS OF MULTIPLIER TO TEST)   4247
19FD  1F          X19FD RAR  ROTATE LO-BIT OF MULTIPLIER INTO THE CARRY BIT           4248
19FE  57          MOV  D,A SAVE REST OF MULTIPLIER BYTE IN D                     W     4249
19FF  79          MOV  A,C PUT HI-ORDER BYTE OF ACCUMULATED PRODUCT IN ACC FOR RIGHT BIT-SHIFT  4250
1A00  D2 08 1A    JNC  X1A08 IF MULTIPLIER BIT WAS 0, DON'T ADD MULTIPLICAND TO ACCUMULATED PRODUCT  4251
1A03  C5          PUSH B SAVE REST OF MULTIPLIER BYTE & BIT-LOOP CTR ON STACK     U     4252
1A04  11 00 00    LXI  D,X0000 LOAD DE WITH LO-16 BITS OF MULTIPLICAND                 4253
1A07  19          DAD  D ADD LO-16 BITS OF MULTIPLICAND TO ACCUMULATED PRODUCT         4254
1A08  C1          X1A08 POP  B FETCH REST OF MULTIPLIER BYTE INTO D, BIT-LOOP CTR INTO E FROM STACK  4255
1A09  CE 00       ACI  H'00' ADD HI-8 BITS OF MULTIPLICAND TO ACCUMULATED PRODUCT     N     4256
1A0B  1F          X1A0B RAR  }                                                        4257
1A0C  4F          MOV  C,A   } SHIFT 3 BYTES OF ACCUMULATED PRODUCT & LOST-BIT BYTE   0     4258
1A0D  7C          MOV  A,H   } RIGHT 1 PLACE                                          4259
```

C D E B

C H L B (AFTER XCHG 19FA)

LO-16 BITS — AFTER LXI 1AB4, AFTER DAD 1AB7

CY H L

HI-8 BITS — AFTER ACI 1AB9

```
1A0E  1F          RAR
1A0F  67          MOV  H,A
1A10  7D          MOV  A,L      SHIFT 3 BYTES OF ACCUMULATED PRODUCT + LOST-BIT
1A11  1F          RAR           BYTE RIGHT 1 PLACE
1A12  6F          MOV  L,A
1A13  7B          MOV  A,B
1A14  1F          RAR
1A15  47          MOV  B,A
1A16  1D          DCR  E        DECREMENT BIT-LOOP CTR
1A17  7A          MOV  A,D      PUT REST OF MULTIPLIER BYTE IN ACC
1A18  C2 FD 19    JNZ  X19FD    LOOP UNTIL ALL 8-BITS OF MULTIPLIER HAVE BEEN USED
1A1B  EB          XCHG          PUT 2-LO ORDER BYTES BACK INTO DE - RESULT OF MULTIPLY (MANTISSA) IS
X1A1C E1          POP  H        FETCH PTR TO MULTIPLIER FROM STACK
1A1D  C9          RET           LOOP TO 19F3 TWO TIMES, THEN JMP NORMALIZE # ROUTINE 18ED, DONE

X1A1E 43          MOV  B,E      8 BIT SHIFT RIGHT
1A1F  5A          MOV  E,D      Acc
1A20  51          MOV  D,C      ØØ → C → D → E → B
1A21  4F          MOV  C,A
1A22  C9          RET

1A23  CD 36 1B    X1A23 CALL X1B36  MOVE SNG-PREC # FROM HOLDING AREA ONTO STACK (1ST #)
1A26  21 9C 1F          LXI  H,X1F9C  LOAD HL WITH ADDR OF SNG-PREC CONSTANT 10 DECIMAL } 2ND #
1A29  CD 43 1B          CALL X1B43  MOVE CONSTANT INTO HOLDING AREA
1A2C  C1          X1A2C POP  B      MOVE FLOATING-PT # FROM STACK INTO BCDE, B = EXPONENT
1A2D  D1                POP  D      (1ST #)

1A2E  EF          X1A2E RST  5      TEST DIVISOR STORED IN HOLDING AREA
1A2F  CA AC 04          JZ   X04AC  IF DIVISOR IS ZERO, PRINT ERROR MESSAGE "DIVISION BY ZERO"
1A32  2E FF             MVI  L,H'FF' SET L FOR DIVIDE COMPUTATION (DIFFERENCE OF EXPONENTS)
1A34  CD AB 1A          CALL X1AAB  COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS
1A37  34                INR  M      ADD TWO TO EXPONENT OF RESULT (LOC Ø415)
1A38  34                INR  M
1A39  2B                DCX  H      BACK-UP PTR TO MSB OF DIVISOR
1A3A  7E                MOV  A,M    STORE MSB OF DIVISOR IN SBI INSTRUCTION AT 1A5C
1A3B  32 5D 1A          STA  X1A5D
1A3E  2B                DCX  H      BACK UP PTR TO MID-BYTE OF DIVISOR
1A3F  7E                MOV  A,M    STORE MID-BYTE OF DIVISOR IN SBI INSTRUCTION AT 1A58
1A40  32 5C 1A          STA  X1A5C
1A43  2B                DCX  H      BACK UP PTR TO LO-BYTE OF DIVISOR
1A44  7E                MOV  A,M    STORE LO-BYTE OF DIVISOR IN SUI INSTRUCTION AT 1A54
1A45  32 58 1A          STA  X1A58
1A48  41                MOV  B,C    DIVIDEND MANTISSA ORIGINALLY IN CDE
1A49  EB                XCHG        MOVE MANTISSA TO B HL.
1A4A  AF                XRA  A      CLEAR ACC
1A4B  4F                MOV  C,A    CLEAR CDE - USED TO ACCUMULATE QUOTIENT
1A4C  57                MOV  D,A
1A4D  5F                MOV  E,A
1A4E  32 63 1A          STA  X1A63  CLEAR 1A63 - EXTRA BYTE OF DIVIDEND AT HI END - ie DIVIDEND IS 2 IN
X1A51 E5          X1A51 PUSH H      PUT DIVIDEND ON STACK (DIVIDEND IN B HL) (WHAT'S LEFT OF IT AT THIS POINT)
1A52  C5                PUSH B
1A53  7D                MOV  A,L    LO-BYTE
1A54  D6 00             SUI  H'00'
1A56  6F                MOV  L,A
1A57  7C                MOV  A,H    MID-BYTE
1A58  DE 00             SBI  H'00'
1A5A  67                MOV  H,A
1A5B  78                MOV  A,B    HI-BYTE
1A5C  DE 00             SBI  H'00'
1A5E  47                MOV  B,A
1A5F  3E 00             MVI  A,H'00' LOAD ACC WITH OVERFLOW BYTE OF DIVIDEND
1A61  DE 00             SBI  H'00'  SUBTRACT CARRY BIT (OVERFLOW FROM HI BYTE OF DIVIDEND)
1A63  3F                CMC         COMPLEMENT CARRY BIT (FOR CONVENIENCE IN SETTING UP THE JMP-ON-CONDITION)

COMPUTE DIVIDEND = DIVIDEND - DIVISOR
                   (in BHL)   (in BHL)  (in 1A5D 1A59 1A55)
IF NC, DIVIDEND WAS ≥ DIVISOR - SUB OK, PUT A 1 IN QUOTIENT
IF C, DIVIDEND WAS < DIVISOR - SUB NOT OK, PUT A Ø IN QUOTIENT
```

SINGLE-PRECISION DIVIDE

(# IN BCDE) ÷ (# IN HOLDING AREA)
(1ST #) (2ND #)

1A60 B H L

PERFORM 8 BIT SHIFT RIGHT ON # IN CDE, B IS LOST-BIT BYTE

IN CDE, LOST-BIT BYTE IS B

AFTER NORMALIZE

```
1A64  D2 6E 1A          JNC  X1A6E JMP IF SUBTRACT NOT LEGITIMATE
1A67  32 60 1A          STA  X1A60 IF SUBTRACT WAS LEGITIMATE, STORE OVERFLOW BYTE BACK INTO MEMORY
1A6A  F1                POP  PSW } REMOVE (DIVIDEND BEFORE SUBTRACT) FROM STACK
1A6B  F1                POP  PSW }
1A6C  37                STC      SET CARRY TO PUT A 1 IN QUOTIENT
1A6D  D2 C1 E1    POP B POP H JNC  XE1C1 DUMMY PUT (DIVIDEND BEFORE SUBTRACT) BACK INTO BHL - CANCELS SUBTRACT IN
1A70  79                MOV  A,C } TEST MSB OF QUOTIENT, IF MSB IS A 1, DIVISION IS COMPLETED TO
1A71  3C                INR  A   } MAXIMUM PRECISION (24 BITS OF QUOTIENT)
1A72  3C                DCR  A   }
1A73  1F                RAR      PUT LAST BIT (CARRY OR NO CARRY FROM SUBTRACT) IN MSB OF ACC FOR
1A74  FA 23 19  ALMOST DONE JM   X1923 IF MSB OF QUOTIENT IS A 1, DIVISION LOOP COMPLETE, PERFORM ROUNDOFF,
1A77  17                RAL      PUT LAST BIT BACK INTO CARRY BIT TO SHIFT IT INTO THE 3 QUOTIENT
1A78  78                MOV  A,E } LO-BYTE
1A79  17                RAL      }
1A7A  5F                MOV  E,A }
1A7B  7A                MOV  A,D } MID-BYTE
1A7C  17                RAL      }
1A7D  57                MOV  D,A }
1A7E  79                MOV  A,C } HI-BYTE
1A7F  17                RAL      }
1A80  4F                MOV  C,A }
1A81  29                DAD  H   } MULTIPLY WHAT'S LEFT OF THE DIVIDEND BY 2
1A82  78                MOV  A,B } (SAME AS SHIFTING THE DIVISOR RIGHT FOR THE NEXT SUBTRACT)
1A83  17                RAL      }
1A84  47                MOV  B,A }
1A85  3A 60 1A          LDA  X1A60 } THIS IS PART OF THE DIVIDEND, MULTIPLY IT BY 2 ALSO
1A88  17                RAL        }
1A89  32 60 1A          STA  X1A60 }
1A8C  79                MOV  A,C } TEST QUOTIENT, IF ENTIRE QUOTIENT IS ZERO
1A8D  B2                ORA  D   } ADJUST EXPONENT BY 1 (PARTIAL NORMALIZATION)
1A8E  B3                ORA  E   } IF BIT SHIFTED INTO QUOTIENT WAS Ø, & ENTIRE QUOTIENT IS Ø, THEN LEFT SHIFT MINUTIA (IF NXT BIT IS 1, IT WILL BE THE MOST
1A8F  C2 51 1A          JNZ  X1A51 JMP IF QUOTIENT IS NOT ØØ ØØ ØØ                                  SIGNIFICANT BIT, SO DECREMENT EXPONENT TO
1A92  E5                PUSH H SAVE 2 BYTES OF DIVIDEND ON STACK                                    COMPENSATE
1A93  21 15 04          LXI  H,X0415 } DECREMENT EXPONENT OF RESULT BY 1
1A96  35                DCR  M       }
1A97  E1                POP  H FETCH THE 2 BYTES OF DIVIDEND FROM STACK
1A98  C2 51 1A          JNZ  X1A51 IF EXPONENT BYTE IS NOT ZERO, JMP TO 1A51 TO CONTINUE DIVISION
1A9B  C3 3E 19          JMP  X193E IF EXPONENT BYTE IS ZERO, PRINT ERROR MESSAGE "OVERFLOW"
-------------------------------------------------------------------------------------------------------------------------------------------------
1A9E  3E FF       X1A9E MVI  A,H'FF' PUT FF IN ACC FOR DIVIDE COMPUTATION (DIFFERENCE OF EXPONENTS)
1AA0  2E AF       XRA A MVI  L,H'AF' DUMMY PUT ØØ IN ACC FOR MULTIPLY COMPUTATION (SUM OF EXPONENTS)
1AA2  21 1E 04          LXI  H,X041E LOAD HL WITH ADDR OF MSB OF 2ND #.
1AA5  4E                MOV  C,M PUT MSB OF 2ND # IN C
1AA6  23                INX  H ADVANCE PTR
1AA7  AE                XRA  M PUT EXPONENT BYTE OF 2ND # IN ACC (FOR MULTIPLY) OR EXPONENT (FOR DIVIDE)
1AA8  47                MOV  B,A SAVE THIS BYTE IN B
1AA9  2E 00             MVI  L,H'00' CLEAR L, NEXT ROUTINE WILL COMPUTE (Ø415) ± (Ø41F)
-------------------------------------------------------------------------------------------------------------------------------------------------
1AAB  78          X1AAB MOV  A,B } TEST EXPONENT BYTE OF 1ST # (DIVIDEND OR MULTIPLICAND IN BCDE)
1AAC  B7                ORA  A   } or MULTIPLICAND
1AAD  CA CD 1A          JZ   X1ACD IF DIVIDEND IS ZERO, DIVISION IS FINISHED, RESULT IN HOLDING AREA
1AB0  7D                MOV  A,L PUT L IN ACC (L=ØØ FOR MULTIPLY, L=FF FOR DIVIDE)
1AB1  21 15 04          LXI  H,X0415 LOAD HL WITH ADDR OF EXP BYTE OF 2ND # (DIVISOR IN HOLDING AREA)
1AB4  AE                XRA  M PUT EXPONENT OF MULTIPLIER IN ACC, PUT INVERSE OF DIVISOR IN ACC
1AB5  80                ADD  B ADD EXPONENT OF MULTIPLICAND OR DIVIDEND TO ADD (FOR MULT E1+E2, FOR DIV E1-E2-1)
1AB6  47                MOV  B,A } TEST EXPONENT OF RESULT FOR LEGITIMATE OVERFLOW
1AB7  1F                RAR      }
1AB8  A8                XRA  B   }
1AB9  78                MOV  A,B PUT EXPONENT OF RESULT IN ACC
1ABA  F2 CC 1A          JP   X1ACC IF CROSS-PRODUCT OF CARRY & MSB OF SUM IS ZERO, THEN IF MSB IS Ø - UNDERFLOW, IF MSB IS 1 - OVERFLOW
1ABD  C6 80             ADI  H'80' ADJUST EXPONENT OF RESULT TO CORRECT VALUE
1ABF  77                MOV  M,A SAVE EXPONENT OF RESULT AT Ø415
```

```
R     4320
2     4321
      4322
      4323
7     4324
RA    4325
      4326 IAS] - IASE
      4327
=     4328
      4329
#     4330 ROUNDOFF TEST AT 1926-1927
      4331 NORMALIZATION, AND PUT SIGN BIT INTO RESULT
      4332 BYTES
      4333
      4334
-     4335
      4336
H     4337
      4338
      4339
0     4340
)     4341
      4342
      4343
G     4344
:     4345
      4346
2     4347
      4348
2     4349
3     4350
BQ    4351
      4352
!     4353
5     4354
      4355
BQ    4356
C>    4357
>     4358
./    4359 (# IN Ø4ØE - Ø415) X (# IN Ø418 - Ø41F)
!     4360 (1ST #)  (2ND #)
N     4361
#     4362
      4363
G     4364
,     4365
      4366 COMPUTE EXPONENT FOR MULTIPLY (L=Ø) OR DIVIDE (L=FF)
7     4367
JM    4368 (FORCE VALUE IN HOLDING AREA TO Ø)
      4369
!     4370
      4371
      4372
GE,+E2 4373, FOR DIV E1-E2-1)
      4374
(     4375
      4376
L     4377
      4378
      4379
```

```
1AC0 CA 1 1A 1C 1A      JZ     X1A1C IF EXPONENT OF RESULT IS NOW ZERO - UNDERFLOW, * or / DONE  J-   4380 RESULT IS ZERO
1AC3 CD 73 18           CALL   X1873 RESTORE LEADING 1 IN BOTH ARGS, COMPUTE XNOR OF SIGN BITS  M    4381
1AC6 77                 MOV    M,A SAVE XNOR OF SIGN BITS AT LOC Ø416                           4382
1AC7 2B           X1AC7 DCX    H BACK-UP PTR TO Ø415                                            +    4383
1AC8 C9                 RET    GO-BACK TO MULTIPLY OR DIVIDE ROUTINE                            I    4384
1AC9 EF           X1AC9 RST    5 PERFORM FLOATING-PT SIGN TEST ON VALUE IN HOLDING AREA               4385
1ACA 2F                 CMA    # IN ACC = Ø IF VALUE ≤ Ø  # IN ACC = FE IF VALUE > Ø            /    4386
1ACB E1                 POP    H REMOVE HALF OF SNG-PREC # PUT ON STACK AT 25F7 (1ACD REMOVES OTHER HALF)  4387
1ACC B7           X1ACC ORA    A TEST MSB OF BYTE IN ACC                                        7    4388
1ACD E1           X1ACD POP    H REMOVE RETURN ADDR FROM STACK (FROM CALL 1AAB or CALL 1ACC or CALL 1AC9)  4389
1ACE F2 D2 19           JP     X19D2 FORCE VALUE IN HOLDING AREA TO ZERO (EXP BYTE = Ø), END OF DIVIDE, RESULT  4390 IN HOLDING AREA - UNDERFLOW
1AD1 C3 3E 19           JMP    X193E IF RESULT IS MINUS, PRINT ERROR MESSAGE "OVERFLOW"          C>   4391                - OVER FLOW
1AD4 CD 51 18     X1AD4 CALL   X1851 LOAD FLOATING-PT # FROM HOLDING AREA INTO BCDE, B = EXP BYTE     4392 MULTIPLY FLOATING-PT # IN
1AD7 78                 MOV    A,B } TEST EXPONENT BYTE                                              4393
1AD8 B7                 ORA    A  }                                                             7    4394 HOLDING AREA BY 10 DECIMAL
1AD9 C8                 RZ     RETURN IF FLOATING-PT VALUE IS Ø                                 H    4395 LEAVE RESULT IN HOLDING AREA
1ADA C6 02              ADI    H'02' ADD 2 TO EXPONENT (MULTIPLY FLOATING-POINT # BY 4)         F    4396          (SNG-PREC)
1ADC DA 3E 19           JC     X193E IF CARRY OCCURS, EXPONENT TOO BIG, PRINT ERROR MESSAGE     Z>   4397 "OVERFLOW"
1ADF 47                 MOV    B,A PUT EXPONENT BYTE BACK INTO APPROPRIATE REGISTER              G    4398
1AE0 CD 9C 18           CALL   X189C ADD # IN BCDE TO # IN HOLDING AREA (FORM 5X #, IN HOLDING AREA)  4399
1AE3 21 15 04           LXI    H,X0415 } MULTIPLY # IN HOLDING AREA BY 2 (INCREMENT EXPONENT BY 1)   4400
1AE6 34                 INR    M       } FORMS 10X # IN HOLDING AREA                            4    4401
1AE7 C0                 RNZ    IF EXPONENT BYTE NOT ZERO, MULTIPLY IS OK & FINISHED             Ø    4402
1AE8 C3 3E 19           JMP    X193E IF CARRY OCCURS, EXPONENT TOO BIG, PRINT ERROR MESSAGE "OVERFLOW"  C>  4403
1AEB 3A 14 04     X1AEB LDA    X0414 FETCH BYTE OF # IN HOLDING AREA, MSB & SIGN BIT            !    4404
1AEE FE 2F              CPI    A'/' DUMMY CMA                                                   /    4405          REST OF RST 5
1AF0 17           X1AF0 RAL      ROTATE SIGN BIT INTO ACC                                             4406 FLOATING-PT SIGN TEST
1AF1 9F           X1AF1 SBB    A (ACC) = (ACC) - (ACC) - CARRY } IF FLOATING POINT # IS MINUS ACC = FF  4407
1AF2 C0                 RNZ                                    }          CARRY = 1              Ø    4408
1AF3 3C                 INR    A                               } IF FLOATING POINT # IS POSITIVE ACC = 1  <  4409
1AF4 C9                 RET                                    }          CARRY = Ø              I    4410
1AF5 06 88        X1AF5 MVI    B,H'88' SET EXPONENT TO 88 (= 2^7 => 88H - 81H = 7)                  4411 } CONVERT BYTE IN ACC TO SNG-PREC VALUE
1AF7 11 00 00           LXI    D,X0000 SET 2 LO-ORDER BYTES OF MANTISSA TO ØØØØ                     4412 }
1AFA 21 15 04     X1AFA LXI    H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE                       !    4413 —
1AFD 4F                 MOV    C,A PUT MOST SIGNIFICANT BYTE OF INTEGER TO CONVERT IN C (SIGN BIT IN C)  4414 CONVERT # IN BADE TO SNG-PREC VALUE
1AFE 70                 MOV    M,B PUT EXPONENT BYTE IN Ø415 WHILE NORMALIZING THE MANTISSA          4415
1AFF 06 00              MVI    B,H'00' FORCE LOST-BIT BYTE TO ØØ, FOR NORMALIZE ROUTINE              4416
1B01 23                 INX    H       } SET SIGN OF RESULT BYTE TO POSITIVE (REMEMBER: 1873 USES XNOR'D OF SIGN  4417
1B02 36 80              MVI    M,H'80' }                                  BITS SEE 192F)        6    4418
1B04 17                 RAL    PUT SIGN BIT IN CARRY                                                4419
1B05 C3 EA 18           JMP    X18EA NORMALIZE # (INVERT SIGN BIT + 2'S COMPLEMENT # IN CDEB IF SIGN  C  4420 IS NEGATIVE)
1B08 CD 25 18     X1B08 CALL   X1825 PERFORM SIGN TEST ON VALUE IN HOLDING AREA                 M%   4421 ABS (VALUE IN HOLDING AREA)
1B0B F0                 RP       IF VALUE IS POSITIVE (OR ZERO) RETURN                               4422
1B0C F7           X1B0C RST    6    TEST FOR VALUE TYPE IN HOLDING AREA.                            4423 ↓ NEGATE VALUE IN HOLDING AREA
1B0D FA 1C 1E           JM     X1E1C NEGATE INTEGER VALUE IN HOLDING AREA (CONVERT # TO POSITIVE VALUE)  4424 ↓
1B10 CA 3A 1C           JZ     X1C3A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"    4425
1B13 21 14 04     X1B13 LXI    H,X0414 LOAD HL WITH ADDR OF MSB BYTE OF # IN HOLDING AREA            4426 ↓ NEGATE FLOATING-POINT #
1B16 7E                 MOV    A,M  FETCH MSB BYTE INTO ACC                                        4427 ↓ (CONVERT # TO POSITIVE VALUE)
1B17 EE 80              XRI    H'80' INVERT SIGN BIT                                               4428
1B19 77                 MOV    M,A  STORE BYTE BACK IN HOLDING AREA                                4429
1B1A C9                 RET    END OF ABS-VALUE PROCESSING                                      I  4430
1B1B CD 25 18     X1B1B CALL   X1825 PERFORM SIGN TEST ON VALUE IN HOLDING AREA                 M%   4431 SGN (VALUE IN HOLDING AREA)
1B1E 6F           X1B1E MOV    L,A } CONVERT SIGN TO A 2-BYTE INTEGER VALUE IN HL                    4432
1B1F 17                 RAL      } ROTATE SIGN BIT INTO ACC                                          4433
1B20 9F                 SBB    A } SET HI ORDER BYTE TO FF IF MINUS OR ØØ IF ZERO OR POSITIVE        4434
1B21 67                 MOV    H,A }                                                               4435
1B22 C3 2D 1C           JMP    X1C2D STORE INTEGER VALUE IN HOLDING AREA - SET VALUE TYPE TO C-Ø2   4436
1B25 F7           X1B25 RST    6  TEST FOR VALUE TYPE (LOC Ø3A4) RZ-CHAR RP-SNG or DBL              4437 PERFORM SIGN TEST ON.
1B26 CA 9A 1C           JZ     X1C9A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"   4438 VALUE IN HOLDING AREA
1B29 F2 28 00           JP     X0028 JMP TO RST 5 FLOATING-PT SIGN TEST IF SNG or DBL PRECISION OPERATION } 4439 (INTEGER OR FLOATING-PT)
                                                                                                          RETURN WITH FLAGS & ACC SET FOR FLOATING POINT VALUE SIGN
```

```
182C 2A 12 04          LHLD X0412 LOAD HL WITH INTEGER VALUE IN HOLDING AREA
182F 7C          X182F MOV  A,H  TEST IF INTEGER VALUE IS ØØØØ
1830 B5                ORA  L    +
1831 C8                RZ       RETURN IF IT IS ØØØØ (ACC = ØØ)
1832 7C                MOV  A,H  PUT HI-ORDER BYTE OF INTEGER IN ACC
1833 C3 F0 1A          JMP  X1AF0 SET FLAGS ACCORDING TO SIGN BIT OF BYTE IN ACC (ACC = FF IF MINUS, ACC = 1 IF POSITIVE)

1836 EB          X1836 XCHG     SAVE HL IN DE
1837 2A 12 04          LHLD X0412 PUT 4-BYTE SINGLE-PRECISION # IN HOLDING AREA AT
183A E3                XTHL
183B E5                PUSH H     Ø412 ON STACK, LSB FIRST, EXPONENT BYTE LAST
183C 2A 14 04          LHLD X0414
183F E3                XTHL
1840 E5                PUSH H
1841 EB                XCHG   RESTORE HL FROM DE
1842 C9                RET

1843 CD 54 18    X1843 CALL X1854 FETCH 4 BYTE # FROM MEMORY (HL IS PTR) INTO BCDE
1846 EB          X1846 XCHG     PUT PTR TO CONSTANT IN MEMORY IN DE, PUT MED & LO BYTES (D E) IN HL
1847 22 12 04          SHLD X0412 STORE MED & LO BYTES OF # } STORE 4 BYTE VALUE
184A 60                MOV  H,B } PUT HI BYTE (C) IN L      } IN HOLDING AREA
184B 69                MOV  L,C }
184C 22 14 04          SHLD X0414 STORE HI BYTE & EXP BYTE
184F EB                XCHG   RESTORE PTR TO CONSTANT IN MEMORY TO HL, DE DON'T CARE
1850 C9                RET

1851 21 12 04    X1851 LXI  H,X0412 LOAD HL WITH ADDR OF SNG-PREC OR INTEGER HOLDING AREA
1854 5E          X1854 MOV  E,M  FETCH VALUE IN HOLDING AREA INTO BCDE, B = EXPONENT BYTE
1855 23                INX  H    IF INTEGER, VALUE IN DE, BC DON'T CARE
1856 56          X1856 MOV  D,M  IF ENTER 1856, MOVE 3 BYTES FROM MEMORY INTO BCD
1857 23                INX  H    (USED BY STRING FCN'S & STRING VARIABLE PROCESSING)
1858 4E                MOV  C,M
1859 23                INX  H
185A 46                MOV  B,M
185B 23          X1858 INX  H
185C C9                RET

185D 11 12 04    X185D LXI  D,X0412 LOAD DE WITH ADDR OF HOLDING AREA (SOURCE PTR), HL HAS DEST. ADDR
1860 06 04             MVI  B,H'04' SET # OF BYTES TO MOVE TO SNG-PREC VALUE
1862 C3 6A 18          JMP  X186A MOVE THE # (4 BYTES)
1865 EB          X1865 XCHG EXCHANGE HL & DE — SOURCE & DESTINATION PTRS SWAPPED
1866 3A A4 03    X1866 LDA  X03A4 } LOAD # OF BYTES TO MOVE INTO B REG
1869 47                MOV  B,A  } (VALUE TYPE = # OF BYTES IN VALUE)
186A 1A          X186A LDAX D   FETCH A BYTE
186B 77                MOV  M,A  STORE BYTE
186C 13                INX  D  } ADVANCE PTRS
186D 23                INX  H  }
186E 05                DCR  B  DECREMENT CHAR CTR
186F C2 6A 18          JNZ  X186A LOOP TIL ALL BYTES HAVE BEEN MOVED
1872 C9                RET

1873 21 14 04    X1873 LXI  H,X0414 LOAD HL WITH ADDR OF MOST SIGNIFICANT BYTE OF VALUE IN HOLDING AREA
1876 7E                MOV  A,M  FETCH MOST SIGNIFICANT BYTE INTO ACC
1877 07                RLC    } FORCE SIGN BIT OF MANTISSA TO 1 } EXTRACT SIGN BIT, RESTORE
1878 37                STC    }                                 } LEADING 1 IN MANTISSA
1879 1F                RAR    }                                 } S, 1XXX XXXX → Ø414
187A 77                MOV  M,A  PUT MOST SIGNIFICANT BYTE BACK IN HOLDING AREA
187B 3F                CMC    COMPLEMENT SIGN BIT
187C 1F                RAR    PUT SIGN BIT IN MSB      X S, 1XX XXXX
187D 23                INX  H  } STORE THIS SIGN BYTE AT LOC Ø416
187E 23                INX  H  }
187F 77                MOV  M,A
1880 79                MOV  A,C  FETCH MOST SIGNIFICANT BYTE OF 2ND # INTO ACC
1881 07                RLC    } EXTRACT SIGN BIT, RESTORE LEADING 1 IN MANTISSA
1882 37                STC    }
```

```
4440 SET FLAGS & ACC
4441 FOR SIGN OF INTEGER VALUE
4442
4443
4444
4445 ACC = 1 IF POSITIVE)
4446
4447 SINGLE-PREC # ONTO STACK
4448   HOLDING AREA TO STACK
4449
4450
4451
4452
4453
4454

4455
4456
4457
4458
4459
4460
4461
4462

4463
4464
4465
4466
4467
4468
4469
4470
4471
4472

4473 MOVE N-BYTES (VARIABLE VALUE)
4474
4475
4476
4477 MOVE N CHAR: 1866  1865
4478   SOURCE ADDR: DE  HL
4479   DEST ADDR: HL  DE
4480   (Ø3A4): # of BYTES TO MOVE
4481
4482
4483
4484
4485

4486 EXTRACT SIGN BITS FROM 2
4487
4488 #'S, RESTORE LEADING 1 IN BOTH
4489
4490 #'S (ONE # IN HOLDING AREA,
4491 ONE # IN BCDE)
4492
4493
4494 SUITABLE FOR SNG- & DBL-PREC
4495
4496
4497 ACC HAS XNOR OF SIGN BITS
4498 Ø416 HAS SIGN OF # IN HOLDING
4499   AREA
```

STACK BEFORE: RET ADDR

STACK AFTER: LSB (Ø412), (Ø413) } SNG-PREC FLOATING POINT #; RET. ADDR

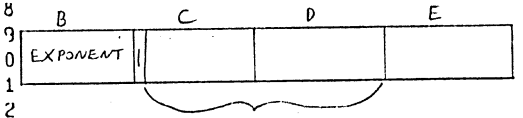
Ø412	(E)	LO BYTE	FETCH INT OR SNG-PREC VALUE FROM MEMORY. STORE IT IN HOLDING AREA
Ø413	(D)	MED BYTE	
Ø414	(C)	HI BYTE	
Ø415	(B)	EXPONENT BYTE	

Ø412	(E)	LOAD BCDE, OR
Ø413	(D)	BCD WITH A
Ø414	(C)	VARIABLE VALUE
Ø415	(B)	

```
                                  RAR   } EXTRACT SIGN BIT, RESTORE LEADING 1 IN MANTISSA  S2 1XXX XXXX      4500
1883  1F                          MOV   C,A PUT MOST SIGNIFICANT BYTE BACK INTO REG C                  O      4501
1884  4F                          RAR   PUT SIGN BIT IN MSB     X S2 1XX XXXX                          4502
1885  1F                          XRA   M FORM XOR OF SIGN BITS IN ACC  S1⊕S2 in MSB of ACC            .      4503
1886  AE                          RET                                                                  I      4504
1887  C9
-------------------------------------------------------------------------------------------------------------------
1888  21 18 04    X1888  LXI   H,X0418 LOAD HL WITH ADDR OF LSB OF DBL-PREC # IN TEMP STORAGE AREA          4505
1888  11 65 18    X1888  LXI   D,X1865 LOAD DE WITH ADDR OF ROUTINE THAT WILL MOVE DBL-PREC #             4506 FROM Ø418 TO Ø4ØE
188E  C3 97 18           JMP   X1897 LOAD DE WITH Ø4ØE, THEN MOVE THE VARIABLE VALUE                C      4507
1891  21 18 04    X1891  LXI   H,X0418 LOAD HL WITH ADDR OF LSB OF DBL-PREC # IN TEMP STORAGE !AREA        4508
1894  11 66 18           LXI   D,X1866 LOAD DE WITH ADDR OF ROUTINE THAT WILL MOVE DBL-PREC # FROM Ø4ØE   4509 TO Ø418.
1897  D5          →X1897 PUSH  D PUT IMPLIED JMP ADDR ON STACK                                        U      4510 ROUTINE :          MOVE #
1898  11 12 04    X1898  LXI   D,X0412 LOAD DE WITH ADDR OF LSB OF # IN HOLDING AREA (INT OR SNG)           4511 1B88 DBL-PREC    Ø418 → Ø4ØE
189B  F7                    RST   6      TEST VALUE TYPE                                                 4512 1B8B #           (HL) → HOLDING AREA
189C  D8                    RC           RETURN IF INT, SNG, OR STRING TYPE                     X        4513 1B91 DBL-PREC    Ø4ØE → Ø418
189D  11 0E 04              LXI   D,X040E LOAD DE WITH ADDR OF LSB OF # IN DBL PREC HOLDING AREA         4514 1B97
18A0  C9                    RET                                                                  I      4515 1B98 LOAD DE WITH LSB ADDR OF #
-------------------------------------------------------------------------------------------------------------------
18A1  78          X18A1  MOV   A,B } TEST EXPONENT BYTE                                                   4516 (HM) - (HR) SIGN TEST (SNG-PREC
18A2  B7                 ORA   A  }                                                              7        4517 MAGNITUDE
18A3  CA 28 00           JZ    X0028 IF EXPONENT IS ZERO, THEN # IN BCDE IS ZERO, DO FLOATING-PT J(SIGN  4518 TEST ON # IN HOLDING AREA TEST)
18A6  21 EF 1A           LXI   H,X1AEF } PUT ADDR 1AEF ON STACK FOR IMPLIED JMP                 !        4519
18A9  E5                 PUSH  H       } (JMP TO ROUTINE TO TEST SIGN BASED ON COMPLEMENT OF BYTE IN ACC)  4520 COMPUTE SIGN OF SNG-PREC
18AA  EF                 RST   5 TEST SIGN OF FLOATING-PT # IN HOLDING AREA                             4521 FLOATING-PT SUBTRACT
18AB  79                 MOV   A,C PUT MSB BYTE OF # IN BCDE IN ACC                                     4522
18AC  C8                 RZ    IF # IN HOLDING AREA IS ZERO, INVERT SIGN BIT, DO SIGN TEST       H       4523 (# IN MEMORY) - (# IN BCDE)
18AD  21 14 04           LXI   H,X0414 LOAD HL WITH ADDR OF MSB BYTE OF # IN HOLDING AREA       !        4524   MINUEND      SUBTRAHEND
18B0  AE                 XRA   M XOR SIGN BITS OF THE 2 # 'S                                    .        4525 ALSO ; MAGNITUDE COMPARE OF
18B1  79                 MOV   A,C PUT MSB OF REGISTER # IN ACC                                         4526 2 FLOATING-POINT #'S
18B2  F8                 RM    IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS COMPLEMENTED SIGN OF SUBTRAHEND  4527 { RZ (#M)=(#R) ACC=Ø  RC (#M)<(#R) ACC=-1
18B3  CD B9 18           CALL  X18B9 TEST THE RELATIVE SIZE OF THE 2 #'S, RC (M)>(R), RNC (M)<(R)  H9   4528 { RNC (#M)≥(#R) ACC=1
18B6  1F          X18B6  RAR   PUT CARRY BIT IN ACC } VERY TRICKY! NOTE: SIGN BITS SAME FOR CALL 18B9  4529 * IF #'S ARE MINUS, MAGNITUDES ARE
18B7  A9                 XRA   C XOR WITH SIGN OF (R #) } REALLY REVERSED DURING MAGNITUDE TEST. IF    4530 (#R) IS MINUS, SIGN OF SUBTRACT
18B8  C9                 RET                            } BASED ON MAGNITUDE OPPOSITE THAT IF I(#R)     4531 IS POSITIVE
-------------------------------------------------------------------------------------------------------------------
18B9  23          X18B9  INX   H  ADVANCE HL TO POINT TO EXPONENT BYTE                          #        4532
18BA  78                 MOV   A,B PUT EXPONENT-BYTE OF REGISTER # IN ACC                               4533 TEST IF SNG-PREC # IN
18BB  BE                 CMP   M  TEST AGAINST EXPONENT BYTE OF # IN MEMORY                    >        4534 BCDE EQUALS SNG-PREC #
18BC  C0                 RNZ      RETURN IF NOT EQUAL                                          @        4535 IN HOLDING AREA (Ø412 - Ø415)
18BD  2B                 DCX   H  } TEST MSB-BYTE, RETURN IF NOT EQUAL                         +        4536
18BE  79                 MOV   A,C }                                                                    4537
18BF  BE                 CMP   M   }                                                           >        4538 IF BOTH #'S ARE SAME SIGN,
18C0  C0                 RNZ       }                                                           @        4539
18C1  2B                 DCX   H  } TEST MID-BYTE, RETURN IF NOT EQUAL                         +        4540 THEN RC # IN MEM > # IN REG
18C2  7A                 MOV   A,D }                                                                    4541
18C3  BE                 CMP   M   }                                                           >        4542      RNC # IN MEM ≤ # IN REG
18C4  C0                 RNZ       }                                                           @        4543      RZ # IN MEM = # IN REG
18C5  2B                 DCX   H  } TEST LO-BYTE, RETURN IF NOT EQUAL                          +        4544
18C6  7B                 MOV   A,E }                                                                    4545
18C7  96                 SUB   M   }                                                                    4546
18C8  C0                 RNZ       }                                                           @        4547
18C9  E1                 POP   H  REMOVE ADDR 18B6 FROM STACK } IF BOTH #'S ARE EQUAL, ZERO             4548
18CA  E1          X18CA  POP   H  REMOVE ADDR 1AEF FROM STACK } FLAG SET, RETURN DIRECTLY TO            4549
18CB  C9                 RET                                 } CALLING ROUTINE ONCE REMOVED  I         4550 (i.e. CALL BEFORE CALL)
-------------------------------------------------------------------------------------------------------------------
18CC  7A          X18CC  MOV   A,D } XOR SIGN BITS OF THE 2 #'S                                          4551
18CD  AC                 XRA   H  }                                                           '        4552 (#HL) - (#DE) SIGN TEST
18CE  7C                 MOV   A,H }                                                                    4553      (INTEGER MAGNITUDE TEST)
18CF  FA F0 18           JM    X18F0 IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS SIGN OF MINUEND (#HL)  4554
18D2  BA                 CMP   D  COMPARE HI-ORDER BYTES (H-D)                                !        4555 RZ  (#HL) = (#DE)  ACC = Ø → =
18D3  C2 F1 18           JNZ   X18F1 IF HI-ORDER BYTES ARE UNEQUAL, SET FLAGS ACCORDINGLY     @        4556 RNC (#HL) ≥ (#DE)  ACC = 1 → >
18D6  7D                 MOV   A,L } COMPARE LO-ORDER BYTES                                             4557 RC  (#HL) < (#DE)  ACC = -1 → <
18D7  93                 SUB   E  }                                                                     4558
18D8  C2 F1 18           JNZ   X18F1 IF LO-ORDER BYTES ARE UNEQUAL, SET FLAGS ACCORDINGLY     @        4559
```

```
18D0  C9                    RET  RETURN WITH ZERO-FLAG SET IF #'S ARE EQU.  ACC=Ø          I    4560
18DC  21 18 04    X18DC     LXI  H,X0418 LOAD HL WITH ADDR OF DBL-PREC VALUE TEMP STORAGE AREA  !    4561
18DF  CC 56 18              CALL X1866 MOVE DBL-PREC # SOURCE ADDR IN DE, DESTINATION ADDR IN HL  M    4562
18E2  11 1F 04    X18E2     LXI  D,X041F LOAD DE WITH ADDR OF EXPONENT BYTE OF 2ND #              4563
18E5  1A                    LDAX D } PUT EXPONENT BYTE IN ACC & TEST IT                           4564  (# Ø4ØE-Ø415) - (# Ø418-Ø41F) SIGN TEST
18E6  B7                    ORA  A }  FLTNG-PT SIGN TEST ON # IN HOLDING AREA                     4565  DBL-PREC MAGNITUDE TEST
18E7  CA 28 00              JZ   X0028 IF EXPONENT IS ZERO, THEN # IN TEMP STORAGE AREA IS ZERO, DO  J(  4566  MINUEND  SUBTRAHEND
18EA  21 EF 1A              LXI  H,X1AEF } PUT ADDR 1AEF ON STACK FOR IMPLIED JMP                 !    4567
18ED  E5                    PUSH H       } (JMP TO ROUTINE TO TEST SIGN BASED ON COMPLEMENT OF BYTE IN ACC)  4568
18EE  EF                    RST  5 TEST SIGN OF FLOATING-PT # IN HOLDING AREA                     4569
18EF  1B                    DCX  D BACK UP PTR TO # IN TEMP STORAGE (POINT TO SIGN BYTE)          4570
18F0  1A                    LDAX D LOAD MSB INTO ACC                                              4571
18F1  4F                    MOV  C,A SAVE MSB OF # IN TEMP STORAGE (2ND #) IN REG C       O       4572
18F2  C8                    RZ   IF # IN HOLDING AREA IS ZERO (1ST #), INVERT SIGN BIT OF 2ND #, DO SIGN TEST  H  4573
18F3  21 14 04              LXI  H,X0414 LOAD HL WITH ADDR OF SIGN BYTE OF # IN HOLDING AREA (1ST #)  4574
18F6  AE                    XRA  M XOR SIGN BITS OF THE 2 #'S                     .               4575
18F7  79                    MOV  A,C PUT MSB OF 2ND # IN ACC                                      4576
18F8  F8                    RM   IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS COMPLEMENTED SIGN OF SUBTRAHEND (2ND #)  4577
18F9  13                    INX  D BACK UP PTR TO 2ND # (POINT TO EXPONENT BYTE)                  4578
18FA  23                    INX  H BACK UP PTR TO 1ST # (POINT TO EXPONENT BYTE)          #        4579
18FB  06 08                 MVI  B,H'08' SET LOOP CTR TO 8 (# OF BYTES IN DBL-PREC #)             4580
18FC  1A          X18FD     LDAX D FETCH A BYTE FROM 2ND #                                        4581
18FE  96                    SUB  M SUBTRACT A BYTE FROM 1ST #                                     4582
18FF  C2 86 18              JNZ  X1886 IF #'S ARE UNEQUAL, PUT CARRY BIT IN ACC, XOR WITH SIGN BIT OF 2ND #  B6  4583 (SEE 1BA1 SNG-PREC ROUTINE)
1902  1B                    DCX  D DECREMENT PTR TO 2ND #                                         4584
1903  2B                    DCX  H DECREMENT PTR TO 1ST #                           +             4585
1904  05                    DCR  B DECREMENT LOOP CTR                                             4586
1905  C2 FD 18              JNZ  X18FD LOOP UNTIL ALL 8 BYTES HAVE BEEN COMPARED          B       4587
1908  C1                    POP  B REMOVE ADDR 1AEF FROM STACK - IMPLIED JMP NO LONGER NEEDED  A  4588
1909  C9                    RET  RZ ACC=Ø, #'S ARE EQUAL                                  I       4589
190A  CD E2 18              CALL X18E2 TEST RELATIVE MAGNITUDE OF #'S                     M       4590  DBL MAGNITUDE TEST
190D  C2 EF 1A              JNZ  X1AEF JMP TO SET FLAGS & ACC, RNC (# Ø4ØE-Ø415) ≥ (# Ø418-Ø41F) ACC=1B  4591  (# Ø4ØE-Ø415) - (# Ø418-Ø41F)
1910  C9                    RET  RZ (# Ø4ØE-Ø415) = (# Ø418-Ø41F) ACC=Ø   RC (# Ø4ØE-Ø415) < (# Ø418-Ø41F) ACC=FF  I  4592
1911  F7          X1C11     RST  6      TEST TYPE OF VALUE IN HOLDING AREA                        4593
1912  2A 12 04              LHLD X0412  LOAD HL WITH INTEGER VALUE FROM HOLDING AREA       *       4594  CINT
1915  F8                    RM          RETURN IF VALUE IS ALREADY INTEGER FORMAT                 4595
1916  CA 3A 1C              JZ   X1C3A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"  4596
1919  D4 40 1C              CNC  X1C40 IF VALUE IS DBL-PREC, CONVERT TO SNG-PREC          TM      4597  CONVERT FLOATING-PT # TO INTEGER
191C  21 3E 19              LXI  H,X193E } PUT ADDR OF ROUTINE TO PRINT ERROR MESSAGE "OVERFLOW"  !>  4598  FORMAT
191F  E5                    PUSH H       } ON STACK                                               4599
1920  3A 15 04    X1C20     LDA  X0415 LOAD ACC WITH EXPONENT BYTE OF # IN HOLDING AREA   :       4600
1923  FE 90                 CPI  H'90' TEST EXPONENT BYTE                                         4601
1925  D2 36 1C              JNC  X1C36 IF EXPONENT IS 15. OR LARGER, TEST IF SPECIAL CASE -32768. IF ITR6 IS,  4602 THEN PUT IT IN INTEGER FORMAT,
1928  CD 8F 1C              CALL X1C8F CONVERT FLOATING-POINT VALUE TO INTEGER FORMAT      M      4603 OTHERWISE PRINT "OVERFLOW"
192B  EB                    XCHG PUT CONVERTED VALUE IN HL                                        4604
192C  D1          X1C2C     POP  D  REMOVE 1 ADDR FROM STACK                              O       4605
192D  22 12 04    X1C2D     SHLD X0412 STORE INTEGER VALUE IN HOLDING AREA  FROM 2173 STORE # TO PRINT IN Ø412 & Ø413  4606
1930  3E 02                 MVI  A,H'02' } STORE Ø2 IN Ø3A4 } SET Ø3A4 TO INTEGER FORMAT  >       4607
1932  32 A4 03    X1C32     STA  X03A4 } SET VALUE TYPE TO INTEGER                        2%      4608
1935  C9                    RET  END OF VALUE TYPE CONVERSION                             I       4609
1936  01 90 90    X1C36     LXI  B,X9080 } PUT SNG-PREC # = -32768 DECIMAL IN BCDE                4610
1939  11 00 00              LXI  D,X0000 }                                                        4611
193C  CD A1 1B              CALL X1BA1 TEST IF # IN HOLDING AREA IS = -32768               M!     4612
193F  C0                    RNZ  IF # ≠ -32768, LEAVE IT IN SNG-PREC FORMAT               @      4613
1940  61                    MOV  H,C } PUT 8000 IN HOLDING AREA (INTEGER FORMAT FOR -32768)        4614
1941  6A                    MOV  L,D }                                                            4615
1942  C3 2C 1C              JMP  X1C2C STORE # INTO HOLDING AREA AS AN INTEGER             C,     4616
1945  F7          X1C45     RST  6      TEST TYPE OF VALUE IN HOLDING AREA                        4617  CSGN
1946  E0                    RPO         RETURN IF VALUE IS ALREADY SINGLE PRECISION FORMAT        4618
1947  FA 60 1C              JM   X1C60 IF VALUE IS INTEGER, CONVERT TO SINGLE PRECISION           4619
```

```
1C4A  CA 8A 1C          JZ    X1C8A IF VALUE IS STRING TYPE, PRINT "TYPE MISMATCH" ERROR MESSAGE   J    4620
1C4D  CD 51 18  X1C40   CALL  X1851 PUT 4-BYTE SINGLE PRECISION VALUE IN BCDE, B = EXPONENT         MQ   4621
1C50  CD 83 1C          CALL  X1C83 SET VALUE TYPE TO SINGLE PRECISION                             M    4622
                                                                                                        4623   SNG-PREC
1C53  78                MOV   A,B } PUT EXPONENT BYTE IN ACC                                       7    4624  Ø415 Ø414 Ø413 Ø412 Ø411
1C54  B7                ORA   A   } TEST EXPONENT                                                 M    4625  EXP   MSB
1C55  C8                RZ    IF EXPONENT IS ZERO, VALUE IS ALSO EXACTLY ZERO, NO MORE CONVERSION M    4626
1C56  CD 73 18          CALL  X1873 RESTORE LEADING 1 IN # IN MSB, EXTRACTED SIGN BIT IN LOC Ø416  M    4627
1C59  21 11 04          LXI   H,X0411 } USE Ø411 CONTENTS AS THE LOST BIT BYTE FOR ROUND-OFF       !    4628
1C5C  46                MOV   B,M     }                                                            F    4629
1C5D  C3 22 19          JMP   X1922 PERFORM ROUNDOFF & NORMALIZE RESULT                            C"   4630
1C60  2A 12 04  X1C60   LHLD  X0412 LOAD HL WITH 2-BYTE INTEGER VALUE FROM HOLDING AREA            *    4631  CONVERT INTEGER TO SINGLE
1C63  CD 83 1C  X1C63   CALL  X1C83 SET VALUE TYPE TO SINGLE PRECISION                             M    4632  PRECISION FORMAT
1C66  7C                MOV   A,H } PUT # TO CONVERT IN ACC & D REG                                 U    4633  (INTEGER VALUE IS IN HL)
1C67  55                MOV   D,L }                                                                     4634
1C68  1E 00             MVI   E,H'00' CLEAR REG E AS PART OF FLOATING-PT #                              4635
1C6A  06 90             MVI   B,H'90' SET VALUE FOR EXPONENT TO 9Ø (=2^15 -> 9ØH-81H=F)                 4636
1C6C  C3 FA 1A          JMP   X1AFA CONVERT # IN BADE TO SNG-PREC (CONSIDER SIGN-BIT + USE NORMALIZE C ROUTINES)  4637
1C6F  F7        X1C6F   RST   6   TEST TYPE OF VALUE IN HOLDING AREA                                P    4638  CDBL
1C70  D0                RNC       RETURN IF VALUE IS ALREADY DOUBLE PRECISION FORMAT                    4639
1C71  CA 8A 1C          JZ    X1C8A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"JH   4640
1C74  FC 60 1C          CM    X1C60 IF VALUE IS INTEGER, CONVERT TO SINGLE PRECISION (1C6Ø)           4641
1C77  21 00 00  X1C77   LXI   H,X0000 } ZERO LO ORDER 4 BYTES OF DOUBLE-PRECISION                !    4642
1C7A  22 0E 04          SHLD  X040E   } HOLDING AREA                                             "    4643
1C7D  22 10 04          SHLD  X0410   }                                                          "    4644
1C80  3E 08             MVI   A,H'08' SET VALUE TYPE TO DOUBLE PRECISION                         >    4645
1C82  21 3E 04          LXI   H,X043E DUMMY MVI A, Ø4. - 1C83 SET VALUE TYPE TO SNG-PRECISION!   >    4646
1C85  C3 32 1C          JMP   X1C32   STORE VALUE TYPE IN Ø3A4, THEN RETURN                            4647
1C88  F7        X1C88   RST   6       TEST VARIABLE TYPE (OR VALUE TYPE IN HOLDING AREA)          H    4648  STRING VALUE CHECK
1C89  C8                RZ            RETURN IF STRING VARIABLE                                        4649
1C8A  1E 0D     X1C8A   MVI   E,H'0D' } PRINT ERROR MESSAGE "TYPE MISMATCH"                       C7   4650
1C8C  C3 B7 04          JMP   X04B7   }                                                                4651  CONVERT FLOATING-POINT VALUE
1C8F  47        X1C8F   MOV   B,A } EXPONENT BYTE IS IN ACC (VALUE IS IN HOLDING AREA)           G    4652  TO INTEGER #, FLOATING POINT
1C90  4F                MOV   C,A } PUT ACC IN ALL 4 REG'S BCDE                                  O    4653  FORMAT ARG (REMOVE THE FRACTION
1C91  57                MOV   D,A } IF EXPONENT IS Ø, VALUE IS Ø, ALL 4 BYTES OF # MUST BE H Ø   _    4654  IN A FLOATING POINT #), RESULT
1C92  5F                MOV   E,A }                                                              7    4655  IS INTEGER FORMAT # IN DE
1C93  B7                ORA   A } TEST ACC - IF EXPONENT IS Ø, BCDE HAVE FLOATING-PT              H    4656
1C94  C8                RZ      } VALUE = Ø  RETURN SINCE # IS A SPECIAL CASE INTEGER                  4657
1C95  E5                PUSH  H SAVE HL ON STACK                                                 MQ   4658
1C96  CD 51 18          CALL  X1851 PUT 4-BYTE SINGLE PRECISION VALUE IN BCDE FROM HOLDING AREA  M    4659
1C99  CD 73 18          CALL  X1873 EXTRACT SIGN BIT (PUT IN Ø416), RESTORE LEADING 1 IN MANTISSA •    4660
1C9C  AE                XRA   M   PUT SIGN BIT OF # IN ACC          -> MANTISSA IS A POSITIVE # AFTER 1C99)  4661
1C9D  67                MOV   H,A SAVE SIGN BIT OF # IN H                                        3    4662
1C9E  FC 83 1C          CM    X1C83 IF # IS NEGATIVE, DECREMENT MANTISSA BY 1 (LSB ROUNDING)        4663  TO CONVERT TO A 16 BIT INTEGER
1CA1  3E 98             MVI   A,H'98' } COMPUTE (FROM EXPONENT BYTE IN B) # OF PLACES TO RIGHT SHIFT TO 4664
1CA3  90                SUB   B       } LEAVE EQUIVALENT INTEGER VALUE IN DE (98 -> 2^23)             4665  ONLY THE BITS IN
1CA4  CD 63 19          CALL  X1963 PERFORM RIGHT SHIFT, # OF PLACES TO SHIFT IN ACC             M    4666
1CA7  7C                MOV   A,H PUT SIGN BYTE IN ACC (See 1C9D)                                      4667
1CA8  17                RAL       TEST SIGN BYTE BY SHIFTING SIGN BIT INTO CARRY BIT                   4668  SAME AS TRUNCATE DOWNWARD, IE INT(-2.3)
1CA9  DC 34 19          CC    X1934 ADD 1 TO RESULT - IF ARG WAS A NEGATIVE # - SINCE RESULT IS A POSITIVE MAGNITUDE AT THE MOMENT, THIS IS THE 4669  = -3
1CAC  06 00             MVI   B,H'00' CLEAR B, THIS IS THE ACCUMULATED LOST-BIT REG, FORCE IT TO Ø FOR 194F  4670
1CAE  DC 4F 19          CC    X194F NEGATE RESULT (INTEGER FORMAT # IN DE) BY SUBTRACTING # FROM Ø  \0   4671
1CB1  E1                POP   H RESTORE HL FROM THE STACK                                        I    4672
1CB2  C9                RET                                                                            4673  IS THE ROUND-DOWN FOR A NEGATIVE #
1CB3  1B        X1C83   DCX   D DECREMENT LO-ORDER BYTE OF 16-BITS OF MANTISSA THAT WILL BECOME AN INTEGER - THIS 4674
1CB4  7A                MOV   A,D } IF DE NOT FF, RETURN                                         #    4675
1CB5  A3                ANA   E   }                                                              <    4676
1CB6  3C                INR   A   }                                                              ≥    4677
1CB7  C0                RNZ       }                                                                   4678
1CB8  0B        X1C88   DCX   B ADJUST EXPONENT BYTE                                             I    4679
1CB9  C9                RET
```



```
1C8A  F7              RST  6     TEST TYPE OF VALUE IN HOLDING AREA
1C8B  F8              RM         RETURN IF VALUE IS ALREADY INTEGER TYPE VALUE
1C8C  D2 DE 1C        JNC  X1CDE IF VALUE IS DBL-PREC TYPE, JMP TO ROUTINE TO EXTRACT INTEGER PORTION OF #
1C8F  CA 8A 1C        JZ   X1C9A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"
1CC2  CD 20 1C        CALL X1C20 IF EXPONENT IS < 2^15, OR VALUE IS -32768, CONVERT USING CINT-DON'T RETURN
X1CC5 21 15 04        LXI  H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE OF # IN HOLDING AREA
1CC8  7E              MOV  A,M   PUT EXPONENT BYTE IN ACC
1CC9  FE 98           CPI  H'98' TEST IF EXPONENT ≥ 23 DECIMAL (ONLY 23 BITS FOR MANTISSA)
1CCB  3A 12 04        LDA  X0412 LOAD ACC WITH LO-ORDER BYTE OF MANTISSA
1CCE  D0              RNC  IF EXPONENT IS 2^23 or LARGER, # CAN ONLY BE AN INTEGER
1CCF  7E              MOV  A,M   PUT EXPONENT BYTE IN ACC
1CD0  CD 8F 1C        CALL X1C8F CONVERT FLOATING-PT VALUE TO AN INTEGER MANTISSA
1CD3  3E 98           MVI  M,H'98' INITIALIZE EXPONENT AT 2^23 FOR NORMALIZATION - see 1CD9)
1CD5  7B              MOV  A,E   } PUT LO-ORDER BYTE OF MANTISSA ON STACK
1CD6  F5              PUSH PSW   }
1CD7  79              MOV  A,C   } PUT SIGN BIT OF RESULT IN CARRY BIT
1CD8  17              RAL        } (USED BY 18EA TO CANCEL CALL 194F AT 1CAE)
1CD9  CD EA 18        CALL X18EA NORMALIZE MANTISSA IN HOLDING AREA
1CDC  F1              POP  PSW   PUT LO-ORDER BYTE OF MANTISSA IN ACC
1CDD  C9              RET

1CDE  21 15 04        X1CDE LXI  H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE OF # IN HOLDING AREA
1CE1  7E              MOV  A,M   PUT EXPONENT BYTE IN ACC
1CE2  FE 90           CPI  H'90' TEST EXPONENT BYTE
1CE4  DA 11 1C        JC   X1C11 IF EXPONENT IS < 90 (< 2^15), # IS < 2^16 - USE CINT ROUTINE
1CE7  C2 FF 1C        JNZ  X1CFF IF EXPONENT IS >90 (>2^15), # IS TO BE CONVERTED USING SHIFT-TECHNIQUE
1CEA  4F              MOV  C,A   SAVE EXPONENT BYTE IN C
1CEB  2B              DCX  H     BACK-UP PTR TO # IN HOLDING AREA
1CEC  7E              MOV  A,M   PUT MSB BYTE IN ACC
1CED  EE 80           XRI  H'80' INVERT SIGN BIT
1CEF  06 06           MVI  B,H'06' INITIALIZE BYTE-LOOP CTR TO 6
X1CF1 2B              DCX  H     BACK-UP PTR TO #
1CF2  B6              ORA  M     OR THE BYTE INTO THE ACC
1CF3  05              DCR  B     DECREMENT BYTE-LOOP CTR
1CF4  C2 F1 1C        JNZ  X1CF1 LOOP UNTIL ALL BYTES OF MANTISSA ORED TOGETHER
1CF7  B7              ORA  A     TEST ACC
1CF8  21 00 80        LXI  H,X8000 LOAD HL WITH 2-BYTE INTEGER FORMAT VALUE = -32768
1CFB  CA 20 1C        JZ   X1C20 IF # IS -32768, STORE RESULT IN HOLDING AREA AS INTEGER VALUE
1CFE  79              MOV  A,C   PUT EXPONENT BYTE IN ACC (see 1CEA)
1CFF  FE 98           X1CFF CPI  H'98' TEST EXPONENT
1D01  D0              RNC  IF EXPONENT IS ≥ 88 (≥2^55), MANTISSA IS ALREADY AN INTEGER, NO CONVERSION NECESSARY
1D02  F5              X1D02 PUSH PSW  SAVE FLAG STATUS ON STACK (CARRY NOT SET IF ENTER AT 1D02 FROM 247C)
1D03  CD 51 1B        CALL X1B51 MOVE HI-4 BYTES OF DBL-PREC # INTO BCDE (USED BY 1B73 ROUTINE)
1D06  CD 73 1B        CALL X1B73 EXTRACT SIGN BIT (PUT IN 0416), RESTORE LEADING 1 IN MANTISSA (MANTISSA IS A POSITIVE # AFTER 1D06)
1D09  AF              XRA  M     PUT SIGN BIT OF # IN ACC
1D0A  2B              DCX  H     BACK-UP HL TO POINT TO LOC 0415
1D0B  36 0A           MVI  M,H'0A' INITIALIZE EXPONENT AT 2^55 FOR NORMALIZATION - see 1D24)
1D0D  F5              PUSH PSW   SAVE SIGN BIT + FLAG STATUS ON STACK
1D0E  FC 27 1D        CM   X1D27 DECREMENT MANTISSA OF # BY 1 - IF # IS MINUS (LSB ROUNDING)
1D11  21 14 04        LXI  H,X0414 LOAD HL WITH ADDR OF MSB BYTE OF #
1D14  3E 08           MVI  A,H'08' } COMPUTE # OF PLACES TO RIGHT SHIFT TO ELIMINATE FRACTION
1D16  90              SUB  B       } IN MANTISSA
1D17  CD 44 1F        CALL X1F44 RIGHT-SHIFT # IN HOLDING AREA n PLACES (n in ACC)
1D1A  F1              POP  PSW   FETCH SIGN BIT + FLAG STATUS FROM STACK
1D1B  FC FC 1E        CM   X1EFC IF # WAS MINUS, TRUNCATION IS DOWNWARD, ADD 1 TO MANTISSA BEFORE
1D1E  AF              XRA  A     } CLEAR THE LOST-BIT BYTE FOR DBL-PREC HOLDING AREA
1D1F  32 00 04        STA  X0400 }
1D22  F1              POP  PSW   FETCH FLAG STATUS FROM STACK
1D23  D0              RNC        RETURN TO 247C IF ENTERED AT 1D02
1D24  C3 90 1E        JMP  X1E90 NORMALIZE MANTISSA IN HOLDING AREA

1D27  21 0E 04        X1D27 LXI  H,X040E LOAD HL WITH ADDR OF LSB BYTE OF MANTISSA OF # IN HOLDING AREA
```

TEST IF MANTISSA = 80 00 00 00 00 00 00 (= -32768, SINCE EXPONENT = 90 = 2^15)

WHEN DONE SHIFTING # IS INTEGER · 2^EXP, 2^88-EXP = INTEGER · 2^88

SIGN RESTORED SINCE MANTISSA IS POSITIVE (see 1D06)

INT

PERFORM INT FUNCTION ON A SINGLE-PRECISION VALUE IN HOLDING AREA

NORMALIZED FORMAT

PERFORM INT FUNCTION ON A DOUBLE-PRECISION VALUE IN HOLDING AREA

SUBTRACT 1 FROM MANTISSA OF # IN HOLDING AREA

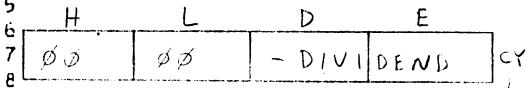
```
102A 7E          X102A  MOV  A,M FETCH A BYTE FROM MANTISSA INTO ACC          4740
102B 35                 DCR  M   DECREMENT THE MANTISSA (WHERE IT IS STORED) 5    4741
102C B7                 ORA  A   TEST THE MANTISSA BYTE                      7    4742
102D 23                 INX  H   ADVANCE PTR TO NEXT BYTE OF MANTISSA        ∅    4743
102E CA 2A 10           JZ   X102A IF BYTE WAS ∅, DECREMENTING PRODUCES A BORROW INTO NEXT HIGHER BYTE J*-  4744 LOOP UNTIL NON-ZERO BYTE FOUND
1031 C9                 RET  DONE                                           I    4745 (# IS NOT ∅ - see ICE4, # HAS MSB - see ID∅6)
                                                                                 4746
1032 E5          X1032  PUSH H  SAVE CONTENTS OF HL ON STACK                      4747 MULTIPLY FOR DIM & ARRAYS
1033 21 A4 03           LXI  H,X03A4 LOAD HL WITH ADDR OF FLAG FOR TYPE OF VALUE IN HOLDING! $AREA  4748
1036 7E                 MOV  A,M FETCH VALUE TYPE INTO ACC                       4749
1037 36 02              MVI  M,H'02' FORCE VALUE TYPE TO 'INTEGER'          6    4750
1039 F5                 PUSH PSW SAVE ORIGINAL VALUE-TYPE ON STACK               4751
103A 60                 MOV  H,B } PUT # OF ELEMENTS IN DIMEN. IN HL             4752      (HL) ← (BC)
103B 69                 MOV  L,C }                                               4753 1075 (HL) = (HL) * (DE)
103C CD 75 10           CALL X1075 PERFORM INTEGER MULTIPLY (MULTIPLICAND IN HL, MULTIPLIER IN DE) M  4754
103F CD 11 1C           CALL X1C11 CONVERT VALUE IN HOLDING AREA TO INTEGER FORMAT  M  4755
1042 F1                 POP  PSW } RESTORE VALUE-TYPE FLAG TO ORIGINAL VALUE     2$   4756
1043 32 A4 03           STA  X03A4 }                                             4757
1046 EB                 XCHG PUT VALUE IN DE                                     4758      (DE) ← (HL)
1047 F1                 POP  H RESTORE HL TO ORIGINAL VALUE FROM STACK           I    4759
1048 C9                 RET                                                      4760
1049 7C                 MOV  A,H PUT SIGN BYTE IN ACC (2ND #)                    4761 INTEGER SUBTRACT
104A 17                 RAL  } TEST SIGN      - MINUS IF ACC = -1                4762
104B 9F                 SBB  A } OF 2ND #       PLUS IF ACC = ∅                  4763 (# IN DE) - (# IN HL)
                  DONE THIS WAY                                          G
                  FOR -32768                                                     4764 (1ST #) - (2ND #)
104C 47                 MOV  B,A SAVE SIGN OF 2ND # IN REG B                M    4765
104D CD 12 1E           CALL X1E12 NEGATE # IN HL (2ND #) BY SUBTRACTING FROM ∅∅∅∅  4766
1050 79                 MOV  A,C PUT ∅∅ IN ACC                                   4767
1051 98                 SBB  B INVERT SIGN OF 2ND # (MINUS = -1, POS = ∅)        CX   4768
1052 C3 58 10           JMP  X1058                                               4769
1055 7C          X1055  MOV  A,H PUT SIGN BYTE IN ACC (2ND #)                    4770 INTEGER ADDITION
1056 17                 RAL  } TEST SIGN      - MINUS IF ACC = -1                4771 (# IN DE) + (# IN HL)
1057 9F                 SBB  A } OF 2ND #       PLUS IF ACC = ∅                  4772      1ST #      2ND #
1058 47          X1058  MOV  B,A SAVE SIGN OF 2ND # IN REG B                G    4773 ARGS  RESULT  CY   SIGN1+SIGN2+CY  XRA H
1059 E5                 PUSH H  PUT 2ND # ON STACK                               4774 + +   +       ∅    ∅                 ∅
105A 7A                 MOV  A,D PUT SIGN BYTE IN ACC (1ST #)                    4775 + −   +       1    ∅                 ∅
105B 17                 RAL  } TEST SIGN      - MINUS IF ACC = -1                4776 − +   −       ∅    −1                ∅
105C 9F                 SBB  A } OF 1ST #       PLUS IF ACC = ∅                  4777 − −   −       1    −1                ∅
105D 19                 DAD  D ADD 1ST # TO 2ND # USING 16-BIT REGISTER ADD - RESULT IN HL  4778
105E 88                 ADC  B } TEST FOR OVERFLOW (ADD PRODUCED A # TOO LARGE TO BE AN INTEGER  4779
105F 0F                 RRC  } BOTH SIGN BITS OF ARGS, CARRY FROM ADD, & SIGN OF RESULT ARE USED
1060 AC                 XRA  H } IN THIS TEST                                    ,
1061 F2 2C 1C           JP   X1C2C IF MAGNITUDE OF RESULT IS SMALL ENOUGH, PUT RESULT IN HOLDING AREA AS INTEGER  4780 RESULT (REMOVES 2 BYTES FROM STACK 1ST)
1064 C5                 PUSH B SAVE SIGN OF 2ND # ON STACK (MINUS IF ACC = -1  PLUS IF ACC = ∅)  E  4781
1065 E9                 XCHG PUT 1ST # IN HL (IT WAS ORIGINALLY IN DE)           4782
1066 CD 63 1C           CALL X1C63 CONVERT # IN HL TO SNG-PREC # IN HOLDING AREA  M  4783
1069 F1                 POP  PSW PUT SIGN OF 2ND # IN ACC FROM STACK             4784
106A E1                 POP  H PUT 2ND # IN HL FROM STACK                        4785
106B CD 36 1B           CALL X1B36 PUT SNG-PREC # IN HOLDING AREA ONTO STACK     M6  4786
106E EB                 XCHG PUT 2ND # IN DE                                     4787
106F CD 2C 1E           CALL X1E2C DO AN ABBREVIATED CONVERSION TO SNG-PREC FORMAT  M,  4788
1072 C3 52 21           JMP  X2152 FETCH 1ST # FROM STACK INTO BCDE, DO A FLOATING-PT ADD  CR!  4789
1075 7C          X1075  MOV  A,H } IF # IN HL IS ∅∅∅∅, STORE INTEGER VALUE ∅∅∅∅ IN HOLDING AREA  4790 INTEGER FORMAT MULTIPLY
1076 95                 ORA  L  } SINCE PRODUCT WILL BE ∅∅∅∅                  S  4791
1077 CA 20 1C           JZ   X1C20 }                                         J-  4792 IF PRODUCT REQUIRES MORE THAN
107A E5                 PUSH H  SAVE 1ST # ON STACK                              4793 2 BYTES - CONVERT TO FLOATING-
107B D5                 PUSH D  SAVE 2ND # ON STACK                          U   4794 POINT & USE FLOATING-POINT
107C CD 06 1E           CALL X1E06 COMPUTE SIGN OF PRODUCT (IN B), CONVERT BOTH #'S TO POSIT MVE  4795 MULTIPLY
107F C5                 PUSH B  SAVE BYTE CONTAINING CORRECT SIGN FOR PRODUCT ON STACK  E  4796
1080 44                 MOV  B,H } PUT 1ST # IN BC                           O   4797  # IN HL - MULTIPLICAND (PUT IN
1081 40                 MOV  C,L }                                           M   4798                    BC AT ID8∅)
1082 21 00 00           LXI  H,X0000 LOAD HL WITH ∅∅∅∅ (START ACCUMULATED VALUE AT ∅∅∅∅)  !  4799  # IN DE - MULTIPLIER
```

```
1085  3E 10        MVI  A,H'10' SET BIT CTR TO 16 DECIMAL (16 BITS TO MULTIPLY)
1087  29           DAD  H       MULTIPLY ACCUMULATED PRODUCT BY 2
1088  DA AD 10     JC   X10AD OVERFLOW - ACCUMULATED PRODUCT TOO BIG FOR 2 BYTES - CONVERT TO FLOATING POINT
1088  EB           XCHG }
108C  29           DAD  H } TEST NEXT BIT OF MULTIPLIER (HI ORDER TO LO ORDER)
108D  EB           XCHG } METHOD: DAD H SAME AS MULTIPLY BY 2; IF CARRY, HI BIT OF MULTIPLIER WAS 1, MULTIPLY ACCUMULATED PRODUCT BY 2, THEN ADD MULTIPLICAND TO PRODUCT
108F  D2 95 10     JNC  X1095 NO CARRY = MULTIPLIER BIT WAS Ø
1091  09           DAD  B   ADD MULTIPLICAND TO ACCUMULATED PRODUCT IF HI BIT OF MULTIPLIER WAS A 1
1092  DA AD 10     JC   X10AD IF CARRY OCCURS, ACCUMULATED PRODUCT IS TOO BIG FOR 2 BYTES - CONVERT TO FLOATING POINT
1095  3D           DCR  A   DECREMENT BIT CTR
1096  C2 87 10     JNZ  X1087 IF BIT CTR IS ØØ, 16 BIT MULTIPLY IS FINISHED, RESULT FITS IN 2 BYTES
1099  C1           POP  B FETCH SIGN-OF-PRODUCT BYTE FROM STACK
109A  D1           POP  D REMOVE 2ND # FROM STACK
109B  7C     X1090 MOV  A,H } TEST HI-ORDER BYTE OF 2 BYTE PRODUCT
109C  B7           ORA  A }
109D  FA A5 10     JM   X10A5 IF PRODUCT HAS HI-ORDER BIT SET - PRODUCT IS 1 BIT TOO LARGE FOR A SIGNED-2-BYTE INTEGER FORMAT
10A0  D1           POP  D REMOVE 1ST # FROM STACK
10A1  78           MOV  A,B PUT SIGN-OF-PRODUCT BYTE IN ACC, PRODUCT IS IN HL
10A2  C3 0E 1E     JMP  X1E0E IF PRODUCT IS MINUS, CONVERT TO POSITIVE USING 2'S COMPLEMENT, THEN STORE PRODUCT AS INTEGER VALUE IN HOLDING AREA
10A5  EE 80  X10A5 XRI  H'80' } TEST IF 2-BYTE PRODUCT IS 8ØØØH = 32768 DECIMAL
10A7  B5           ORA  L }
10A8  CA 9E 10     JZ   X109E JMP IF IT IS, PROCESS THIS SPECIAL CASE
10AB  EB           XCHG PUT 2ND # IN HL (POPPED FROM STACK AT 109A)
10AC  01 C1 E1     LXI  B,XE1C1 DUMMY POP B POP H, PUT PRODUCT SIGN BYTE IN BC, 2ND # IN HL
10AF  CD 63 1C     CALL X1C63 CONVERT INTEGER # IN HL (2ND #) TO FLOATING POINT IN HOLDING AREA AT Ø412
10B2  E1           POP  H PUT 1ST # IN HL
10B3  CD J6 19     CALL X1936 PUT 2ND # FLOATING POINT VALUE ON STACK
10B6  CD 63 1C     CALL X1C63 CONVERT INTEGER # IN HL (1ST #) TO FLOATING POINT IN HOLDING AREA AT Ø412
10B9  C1     X10B9 POP  B } PUT 2ND # IN BCDE - FLOATING POINT VALUE B=EXP BYTE
10BA  D1           POP  D }
10BB  C3 03 19     JMP  X1903 PERFORM FLOATING POINT MULTIPLY
10BE  78     X10BE MOV  A,B } TEST SIGN-OF-PRODUCT BYTE
10BF  B7           ORA  A }
10C0  C1           POP  B REMOVE 1ST # FROM STACK (NO LONGER NEEDED)
10C1  FA 20 1C     JM   X1C20 IF PRODUCT SHOULD BE MINUS, THEN 8ØØØH = -32768 DECIMAL ALREADY, STORE AS INTEGER IN HOLDING AREA
10C4  D5           PUSH D SAVE DE ON STACK (2ND #)
10C5  CD 63 1C     CALL X1C63 CONVERT INTEGER # IN HL TO FLOATING-POINT
10C8  D1           POP  D FETCH DE FROM STACK (2ND #)
10C9  C3 13 19     JMP  X1913 NEGATE FLOATING-POINT # IN HOLDING AREA, RESULT IS +32768 IN HOLDING AREA
-------------------------------------------------------------------------------------------------------------------------------
10CC  7C     X10CC MOV  A,H } TEST DIVISOR (INTEGER FORMAT) IN HL
10CD  B5           ORA  L }
10CE  CA AC 04     JZ   X04AC } IF DIVISOR IS ZERO, PRINT ERROR MESSAGE "DIVISION BY ZERO"
10D1  CD 06 1E     CALL X1E06 COMPUTE SIGN BIT OF RESULT (IN B), CONVERT BOTH ARGS TO POSITIVE INTEGERS
10D4  C5           PUSH B SAVE SIGN-OF-RESULT BYTE ON STACK
10D5  EB           XCHG   SWAP ARGUMENTS - PUT DIVISOR IN DE, PUT DIVIDEND IN HL
10D6  CD 12 1E     CALL X1E12 NEGATE DIVIDEND - IN HL & HOLDING AREA
10D9  44           MOV  B,H } PUT NEGATED DIVIDEND IN BC
10DA  4D           MOV  C,L }
10DB  21 00 00     LXI  H,X0000 CLEAR HL - USED TO COLLECT BITS SHIFTED OUT OF THE DIVIDEND
10DE  3E 11        MVI  A,H'11' SET BIT-LOOP CTR TO 17 (16 BIT DIVISION + 1 FOR OVERFLOW)
10E0  F5           PUSH PSW  SAVE BIT-LOOP CTR ON STACK
10E1  B7           ORA  A    CLEAR THE CARRY BIT
10E2  C3 EF 10     JMP  X10EF START THE LOOP BY SHIFTING THE NEGATED DIVIDEND INTO HL (1 BIT)
10E5  F5     X10E5 PUSH PSW SAVE BIT-LOOP CTR ON STACK
10E6  E5           PUSH H SAVE ACCUMULATED BITS OF DIVIDEND ON STACK
10E7  09           DAD  B SUBTRACT DIVISOR FROM WHAT'S LEFT OF THE DIVIDEND
10E8  D2 EE 10     JNC  X10EE IF NO CARRY, DIVISOR IS LARGER THAN DIVIDEND - SUBTRACT NOT LEGITIMATE
10EB  F1           POP  PSW ELIMINATE BITS OF DIVIDEND FROM STACK - SUBTRACT OK - RESULTS ALREADY IN HL
10EC  37           STC  SUBTRACT IS LEGITIMATE - SET CARRY TO SHIFT A 1 INTO ACCUMULATED QUOTIENT (IN DE)
10ED  3E E1  POP H MVI  A,H'E1' DUMMY RESTORE HL TO VALUE BEFORE SUBTRACT
```

INTEGER DIVIDE - \

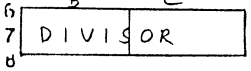
(# IN DE) ÷ (# IN HL)

ARG1 ARG2



SHIFT LEFT 1 BIT AT 1DEF - 1DFA

CY IS 1 IF SUBTRACT OK
CY IS Ø IF SUBTRACT NOT OK



107

```
1DEF  79        X1DEF  MOV  A,E     }  SHIFT 2 BYTES IN DE LEFT 1 PLACE                                  4860
1DF0  17               RAL          }  - MULTIPLY QUOTIENT BY 2 (ALSO PUSH A BIT OF -                   4861
1DF1  5F               MOV  E,A     }  (-DIVIDEND) INTO HL)                                             4862
1DF2  7A               MOV  A,D     }                                                                   4863
1DF3  17               RAL          }                                                                   4864
1DF4  57               MOV  D,A     }                                                          H        4865
1DF5  7D               MOV  A,L     }  SHIFT 2 BYTES IN HL LEFT 1 PLACE                                 4866
1DF6  17               RAL          }                                                                   4867
1DF7  6F               MOV  L,A     }  - MULTIPLY NEGATED DIVIDEND BY 2                                 4868
1DF8  7C               MOV  A,H     }                                                                   4869
1DF9  17               RAL          }                                                                   4870
1DFA  67               MOV  H,A     }                                                                   4871
1DFB  F1               POP  PSW  FETCH BIT-LOOP CTR FROM STACK                                          4872
1DFC  3D               DCR  A    DECREMENT BIT-LOOP CTR                                                 4873
1DFD  C2 E5 1D         JNZ  X1DE5  CONTINUE LOOPING UNTIL ALL 17 BITS OF DIVISION COMPLETED      B     4874
1E00  EB               XCHG  PUT QUOTIENT IN HL, PUT REMAINDER OF (ARG1/ARG2) * ARG2                    4875
1E01  C1               POP  B  FETCH SIGN-OF-RESULT BYTE FROM STACK                              A       4876
1E02  D5               PUSH D  PUT SOMETHING ON STACK TO MATCH POP D AT 1DAB (THIS RESULT WILL NEVER F   4877
1E03  C3 9B 1D         JMP  X1D9B  COMBINE SIGN BIT WITH RESULT - STORE RESULT IN HOLDING AREA   C       4878  BRANCH TO 1DAB) - RESULT - REMAINDER IS PRESERVED IN DE
1E06  7C        X1E06  MOV  A,H  PUT HI-ORDER BYTE OF 1ST # IN ACC  } COMPUTE SIGN BIT OF                  4879
1E07  AA               XRA  D    XOR WITH HI-ORDER BYTE OF 2ND #   } INTEGER PRODUCT                *       4880  FOR INTEGER MULTIPLY, COMPUTE
1E08  47               MOV  B,A  STORE RESULT IN B                                                G       4881  SIGN OF RESULT (XOR OF SIGN OR DIVIDE
1E09  CD 0D 1E         CALL X1E0D  TEST 1ST # IN HL, IF NEGATIVE, CONVERT TO POSITIVE USING 2'S COMPLEMENT      4882  BITS), STORE IN B, CONVERT
1E0C  EB               XCHG  SWAP #'S, 2ND # IN HL                                                      4883
1E0D  7C        X1E0D  MOV  A,H  } TEST # IN HL                                                          4884  BOTH #'S TO POSITIVE INTEGERS
1E0E  B7        X1E0E  ORA  A    } IF # IS NEGATIVE, CONVERT TO POSITIVE USING 2'S COMPLEMENT      7       4885
1E0F  F2 20 1C         JP   X1C20  } IF # IS POSITIVE, JMP, STORE RESULT IN HOLDING AREA          -       4886
1E12  AF        X1E12  XRA  A    CLEAR ACC                                                       /       4887
1E13  4F               MOV  C,A  SAVE 00 IN C                                                   0       4888  NEGATE INTEGER VALUE IN HL
1E14  95               SUB  L    SUBTRACT LO-BYTE OF INTEGER FROM 00                                    4889  BY SUBTRACTING FROM 0000
1E15  6F               MOV  L,A  PUT RESULT BACK IN L                                                   4890
1E16  79               MOV  A,C  PUT 00 IN ACC                                                          4891
1E17  9C               SBB  H    SUBTRACT HI-BYTE OF INTEGER FROM 00 WITH BORROW                        4892
1E18  67               MOV  H,A  PUT RESULT BACK IN H                                                   4893
1E19  C3 20 1C         JMP  X1C20  STORE INTEGER VALUE IN HOLDING AREA - SET VALUE TYPE TO 02   C-      4894
1E1C  2A 12 04  X1E1C  LHLD X0412  LOAD HL WITH INTEGER VALUE FROM HOLDING AREA                  *       4895
1E1F  CD 12 1E         CALL X1E12  NEGATE INTEGER VALUE, STORE BACK INTO HOLDING AREA            M       4896  SPECIAL CASE -(32000) = 8000
1E22  7C               MOV  A,H  } TEST IF INTEGER RESULT OF NEGATION IN 1E12 IS 8000           /       4897  CANNOT STORE NEGATED (-32768) AS
1E23  EE 80            XRI  H'80'}                                                                      4898    32768 IN INTEGER FORMAT
1E25  B5               ORA  L    }                                                              5       4899
1E26  C0        <DONE  RNZ  RETURN IF RESULT ≠ 8000 (-32768)                                    2       4900
1E27  EB               XCHG  PUT INTEGER # IN DE (LO ORDER 2 BYTES OF SNG-PREC #)                       4901
1E28  CD 83 1C         CALL X1C83  SET VALUE TYPE TO SINGLE PRECISION                            M       4902
1E2B  AF               XRA  A    CLEAR ACC                                                      /       4903
1E2C  06 98     X1E2C  MVI  B,H'98'  SET VALUE FOR EXPONENT TO 98 (= 2^23 → 98_H - 81_H = 23_10)         4904
1E2E  C3 FA 1A         JMP  X1AFA  CONVERT # IN BADE TO SNG-PREC (CONSIDER SIGN BIT IN ACC & USE NORMALIZE ROUTINE)  C  4905
1E31  D5        X1E31  PUSH D    SAVE ARG1 ON STACK                                              U       4906
1E32  CD CC 1D         CALL X1DCC  PERFORM INTEGER DIVIDE - (# IN DE) \ (# IN HL) - DE HAS MOD RESULT  ML  4907    MOD
1E35  AF               XRA  A    CLEAR ACC & CARRY BIT                                          /       4908
1E36  82               ADD  D    }                                                                      4909  ARG1 MOD ARG2
1E37  1F               RAR       } PUT REMAINDER TERM (ARG1 MOD ARG2) IN HL                             4910  (# IN DE)    (# IN HL)
1E38  67               MOV  H,A  } RIGHT SHIFT ADJUSTS REMAINDER - COMPENSATES                          4911
1E39  7B               MOV  A,E  } FOR 17th SHIFT IN INTEGER DIVIDE                                     4912  A MOD B = A-[B*A\B]
1E3A  1F               RAR       }                                                                      4913
1E3B  6F               MOV  L,A  }                                                                      4914
1E3C  CD 30 1C         CALL X1C30  SET VALUE TYPE (LOC 03A4) TO INTEGER TYPE (STORE 02)          MD      4915
1E3F  F1               POP  PSW  FETCH ARG1 FROM STACK - SIGN OF RESULT SAME AS SIGN OF ARG1             4916
1E40  C3 0E 1E         JMP  X1E0E  NEGATE RESULT IF ARG1 IS NEGATIVE - STORE RESULT IN HOLDING AREA  S   4917
1E43  21 1F 04  X1E43  LXI  H,X041E  LOAD HL WITH ADDR OF MSB(SIGN) BYTE OF # TO SUBTRACT              4918
1E46  7E               MOV  A,M  PUT SIGN BYTE IN ACC                                                   4919  DOUBLE-PRECISION SUBTRACT
                                                                                                               (# IN 040E-0415) - (# IN 0418-041F
```

```
1E47  EE 80          XRI  H'80'COMPLEMENT SIGN BIT                          } NEGATE DBL-PREC # IN
1E49  77             MOV  M,A PUT SIGN BYTE BACK INTO TEMP STORAGE AREA     } TEMP STORAGE
1E4A  21 1F 04  X1E4A LXI  H,X041F LOAD HL WITH ADDR OF EXPONENT BYTE OF # IN TEMP STORAGE
1E4D  7E             MOV  A,M  } TEST EXPONENT BYTE OF (2ND #)
1E4E  B7             ORA  A    }
                  DONE
1E4F  C8          <- RZ   ADD OR SUB DONE IF # IS Ø, RESULT ALREADY IN HOLDING AREA
                  1# IS ZERO
1E50  47             MOV  B,A SAVE EXPONENT BYTE IN REG B
1E51  2B             DCX  H   } BACK-UP PTR (TO Ø41E)
1E52  4E             MOV  C,M } PUT SIGN BYTE IN REG C. (2ND #)
1E53  11 15 04       LXI  D,X0415 } LOAD ACC WITH EXPONENT BYTE OF 1ST #
1E56  1A             LDAX D    }
1E57  B7             ORA  A  TEST EXPONENT BYTE
                  DONE
1E58  CA 8A 1B    <- JZ   X1B8A ADD OR SUB DONE IF # IS Ø, STORE RESULT IN HOLDING AREA
                  OTHER # IS ZERO
1E5B  90             SUB  B SUBTRACT EXPONENT BYTES TO TEST RELATIVE SIZE OF #'S
1E5C  D2 76 1E       JNC  X1E76 JMP IF (1ST # IN HOLDING AREA) ≥ (2ND # IN TEMP STORAGE)
1E5F  2F             CMA    } FORM 2'S COMPLEMENT OF DIFFERENCE OF EXPONENTS
1E60  3C             INR  A } (POSITIVE # IN ACC AFTERWARDS)
1E61  F5             PUSH PSW SAVE EXPONENT DIFFERENCE ON STACK
1E62  0E 08          MVI  C,H'08' SET BYTE CTR FOR EXCHANGE TO Ø8 (DBL-PREC SIZE)
1E64  23             INX  H ADJUST HL TO POINT TO EXPONENT BYTE OF 2ND #, DE PTS TO EXPONENT BYTE OF 1ST #
1E65  E5             PUSH H SAVE PTR TO 2ND # ON STACK (Ø41F)
1E66  1A       X1E66 LDAX D FETCH A BYTE FROM 1ST # INTO ACC   } SWAP DBL-PREC #'S
1E67  46             MOV  B,M FETCH A BYTE FROM 2ND # INTO B   } RESULT: # AT Ø4ØE....
1E68  77             MOV  M,A STORE BYTE FROM 1ST #            } IS THE LARGER #
1E69  78             MOV  A,B } STORE BYTE FROM 2ND #          }
1E6A  12             STAX D   }
1E6B  1B             DCX  D   } DECREMENT PTRS TO #'S
1E6C  2B             DCX  H   }
1E6D  0D             DCR  C DECREMENT BYTE CTR                 }
1E6E  C2 66 1E       JNZ  X1E66 LOOP UNTIL 8 BYTES HAVE BEEN MOVED }
1E71  E1             POP  H FETCH PTR TO EXPONENT BYTE OF 2ND # IN HL (Ø41F)
1E72  46             MOV  B,M PUT EXPONENT BYTE OF 2ND # ('SMALLER #) IN B } NOTE: THESE 2
1E73  2B             DCX  H   } PUT SIGN + MSB BYTE OF 2ND # IN C          } BYTES SAME AS FOR SNG-PREC # IN BCDE
1E74  4E             MOV  C,M }                                           } FOR 1B73
1E75  F1             POP  PSW FETCH EXPONENT DIFFERENCE FROM STACK
1E76  FE 39    X1E76 CPI  A'9' } TEST IF EXPONENT DIFFERENCE IS 57. OR GREATER
                  DONE
1E78  D0          <- RNC    } IF IT IS, # AT Ø418... TOO SMALL TO AFFECT ANSWER (NOTE: 55 BITS OF VALUE + 1 BIT IMPLIED = 56)
                  1# TOO SMALL
1E79  F5             PUSH PSW SAVE EXPONENT DIFFERENCE ON STACK
1E7A  CD 73 1B       CALL X1B73 COMPUTE XNOR OF SIGN BITS IN ACC, RESTORE LEADING 1'S IN MANTISSA'S - SIGN OF 1ST # IN LOC Ø416
1E7D  23             INX  H ADVANCE PTR TO LOC Ø417
1E7E  36 00          MVI  M,H'00' SET LOST-BIT BYTE TO ØØ (LOST-BIT BYTE FOR Ø418-Ø41F)
1E80  47             MOV  B,A SAVE XNOR OF SIGN BITS IN REG B
1E81  F1             POP  PSW PUT EXPONENT DIFFERENCE INTO ACC FROM STACK
1E82  21 1E 04       LXI  H,X041E LOAD HL WITH ADDR OF MSB OF SMALLER # AT Ø418-Ø41F
1E85  CD 44 1F       CALL X1F44 SHIFT SMALLER MAGNITUDE # IN Ø418-Ø41E RIGHT (# IN ACC) PLACES
1E88  3A 17 04       LDA  X0417 } FETCH LOST-BIT BYTE OBTAINED BY SHIFTING THE SMALLER # RIGHT
1E8B  32 0D 04       STA  X040D } STORE THIS BYTE AT THE LOC (Ø4ØD) FOR LOST-BIT BYTE OF RESULT (USED BY ROUND OFF ROUTINE IEEA)
1E8E  78             MOV  A,B PUT XNOR OF SIGN BITS IN ACC
1E8F  B7             ORA  A TEST BYTE
1E90  F2 A4 1E       JP   X1EA4 JMP IF THE SIGN BITS ARE OPPOSITE
1E93  CD 19 1F       CALL X1F19 ADD ADJUSTED SMALLER # (Ø418-Ø41F) TO # AT Ø4ØE-Ø415, RESULT AT Ø4ØE-MØ415
                  ALMOST DONE
1E96  D2 EA 1E    <- JNC  X1EEA IF NO CARRY, ADDITION DID NOT PRODUCE OVERFLOW, PERFORM ROUND-OFF & NORMALIZE
1E99  EB             XCHG PUT ADDR OF EXPONENT BYTE OF RESULT IN HL
1E9A  34             INR  M ADD 1 TO EXPONENT OF RESULT
                  ERROR
1E9B  CA 9E 19    <- JZ   X193E IF EXPONENT IS NOW ZERO, PRINT ERROR MESSAGE "OVERFLOW"
1E9E  CD 70 1F       CALL X1F70 SHIFT MANTISSA OF RESULT IN Ø4ØE-Ø414 RIGHT 1 PLACE, OVERFLOW BIT IN CARRY SHIFTED INTO HI-ORDER BIT OF MANTISSA
1EA1  C3 EA 1E       JMP  X1EEA PERFORM ROUNDOFF & NORMALIZE RESULT
1EA4  3E 9E    X1EA4 MVI  A,H'9E' } LOAD ACC WITH OP CODE FOR SBB M
1EA6  CD 18 1F       CALL X1F18 } COMPUTE DIFFERENCE OF THE 2 MANTISSA'S
1EA9  23             INX  H ADVANCE PTR TO LOC Ø42Ø

                                                                                  4920
                                                                                  4921
                                                                                  4922
                                                                                  4923 DOUBLE-PRECISION ADD
                                                                                  4924   (1ST #) ± (2ND #)
                                                                                  4925   Ø4ØE...Ø415   Ø418...Ø41F
                                                                                  4926
                                                                                  4927
                                                                                  4928
                                                                                  4929
                                                                                  4930
                                                                                  4931
                                                                                  4932 (MOVE 8 BYTES Ø418.. -> Ø4ØE)
                                                                                  4933
                                                                                  4934
                                                                                  4935
                                                                                  4936
                                                                                  4937
                                                                                  4938
                                                                                  4939
                                                                                  4940
                                                                                  4941
                                                                                  4942
                                                                                  4943
                                                                                  4944
                                                                                  4945
                                                                                  4946
                                                                                  4947
                                                                                  4948
                                                                                  4949
                                                                                  4950
                                                                                  4951
                                                                                  4952
                                                                                  4953
                                                                                  4954
                                                                                  4955
                                                                                  4956
                                                                                  4957
                                                                                  4958
                                                                                  4959
                                                                                  4960
                                                                                  4961
                                                                                  4962
                                                                                  4963
                                                                                  4964
                                                                                  4965
                                                                                  4966
                                                                                  4967
                                                                                  4968
                                                                                  4969
                                                                                  4970
                                                                                  4971
                                                                                  4972
                                                                                  4973
                                                                                  4974
                                                                                  4975
                                                                                  4976
                                                                                  4977
                                                                                  4978
                                                                                  4979
```

```
1EAA 7E          MOV  A,M   } COMPLEMENT LOC Ø41F                                  4980
1EAB 2F          CMA         (CANCELLED BY SAME SEQUENCE AT 1F31 - 1F33 IF CALL AT 1EAD IS EXECUTED)  4981
1EAC 77          MOV  M,A   }                                                      4982
1EAD DC 31 1F    CC   X1F31 IF CARRY SET (OVERFLOW = WRONG SIGN ON #) - CHANGE SIGN OF RESULT (STORED AT  \1  4983
1EB0 AF    X1EB0 XRA  A      CLEAR ACC (THIS LOOP ALLOWS 8 1-BYTE (8-BIT) LEFT SHIFTS AS PART  /OF  4984 NORMALIZING
1EB1 47    X1EB1 MOV  B,A    SAVE SHIFT CTR IN B (COUNTS # OF BIT SHIFTS - USED TO ADJUST EXPONENT  NT  4985
1EB2 3A 14 04    LDA  X0414 } PUT HI-ORDER BYTE OF # TO BE NORMALIZED IN ACC + TEST IT  : BYTE)  4986
1EB5 B7          ORA  A     }                                                  7     4987
1EB6 C2 D9 1E    JNZ  X1ED9 IF HI-ORDER BYTE IS NOT ZERO, THEN FULL-BYTE SHIFT NOT APPLICABLE,  BY  JMP TO SINGLE BIT SHIFT ROUTINE  4988
1EB9 21 0D 04    LXI  H,X040D LOAD HL WITH ADDR OF LOST-BIT BYTE OF RESULT (BYTE BELOW HOLDING  AREA)  4989
1EBC 0E 08       MVI  C,H'08' SET BYTE-LOOP CTR TO Ø8 (56 BITS OF MANTISSA + LOST BIT BYTE)  4990
1EBE 56    X1EBE MOV  D,M FETCH BYTE FROM MANTISSA INTO REG D                  V     4991
1EBF 77          MOV  M,A STORE ACC INTO MANTISSA                                  4992
1EC0 7A          MOV  A,D MOVE BYTE FROM MANTISSA INTO ACC                         4993
1EC1 23          INX  H ADVANCE PTR TO MANTISSA                          Ø         4994
1EC2 0C          DCR  C DECREMENT BYTE-LOOP CTR                                    4995
1EC3 C2 BE 1E    JNZ  X1EBE LOOP UNTIL ALL 8 BYTES HAVE BEEN SHIFTED     B>        4996
1EC6 78          MOV  A,B PUT SHIFT CTR IN ACC FROM B                              4997
1EC7 D6 08       SUI  H'08' SUBTRACT 8 (EACH BYTE SHIFT SAME AS 8 BIT SHIFTS, SUBTRACT 8 FROM  V EXPONENT FOR EACH SHIFT)  4998
1EC9 FE C0       CPI  H'C0' TEST IF 8 BYTE SHIFTS HAVE BEEN DONE YET (64 BIT SHIFTS)  @  4999
1ECB C2 B1 1E    JNZ  X1EB1 JMP TO 1EB1 IF MORE SHIFTS TO DO, OTHERWISE ALL 8 BYTES WERE FOUND TO  B1  5000 BE Ø. THEREFORE RESULT IS Ø
1ECE C3 02 19    JMP  X1902 SET EXPONENT BYTE TO ZERO, RESULT IN HOLDING AREA  C  5001
                 (TOO SMALL)
1ED1 05    X1ED1 DCR  B     DECREMENT SHIFT CTR (EACH SHIFT SAME AS MULTIPLY BY 2, SUBTRACT 1  5002 FROM EXPONENT TO COMPENSATE)
1ED2 21 0D 04    LXI  H,X040D LOAD HL WITH ADDR OF LOST-BIT BYTE OF RESULT IN HOLDING  !AREA  5003
1ED5 CD 78 1F    CALL X1F78 SHIFT 56 BIT MANTISSA + LOST-BIT BYTE LEFT 1 BIT  M  5004
1ED8 B7          ORA  A     TEST SHIFT CTR                                    7  5005
1ED9 F2 D1 1E    JP   X1ED1 IF MSB OF # IS NOT A 1, THEN JMP TO 1ED1 TO DO A 1 BIT LEFT SHIFT  Q  5006
1EDC 78    X1ED9 MOV  A,B   PUT SHIFT CTR IN ACC FROM B                          5007
1EDD B7          ORA  A     TEST SHIFT CTR                                    7  5008
1EDE CA EA 1E    JZ   X1EEA IF SHIFT CTR IS ZERO, PERFORM ROUNDOFF, RESULT STORED IN HOLDING  J AREA  5009
1EE1 21 15 04    LXI  H,X0415 LOAD HL WITH ADDR OF EXPONENT OF RESULT  !  5010
1EE4 86          ADD  M     COMPUTE EXPONENT = EXPONENT - (# OF SHIFTS TO NORMALIZE)  5011
1EE5 77          MOV  M,A   PUT BYTE BACK INTO RESULT                         5012
1EE6 D2 02 19    JNC  X1902 IF NO CARRY, NORMALIZED # IS SO SMALL THAT IT IS SAME AS Ø  5013
                 (DONE)
1EE9 C8          RZ   IF EXPONENT BYTE IS ZERO, NORMALIZED # SAME AS Ø, ADD IS DONE  H  5014
                 (DONE)
1EEA 3A 0D 04    LDA  X040D LOAD ACC WITH LOST-BIT BYTE OF RESULT  !  5015
1EED B7    X1EED ORA  A     TEST LOST BIT (MSB OF ACC)                        7  5016
1EEE FC FC 1E    CM   X1EFC IF BIT SHIFTED OUT OF SMALLER MANTISSA WAS A 1, ROUND UP BY ADDING 1  !  5017 TO THE # AT Ø4ØE ... Ø414
1EF1 21 16 04    LXI  H,X0416 } PUT BYTE WITH SIGN OF LARGER # IN ACC  !  5018
1EF4 7E          MOV  A,M     }   (or # IN HOLDING AREA)                          5019
1EF5 E6 80       ANI  H'80' ELIMINATE ALL BUT THE SIGN BIT                         5020
1EF7 2B          DCX  H }                                                  +       5021
1EF8 2B          DCX  H } BACK-UP PTR TO MSB OF MANTISSA Ø414              +       5022
1EF9 AE          XRA  M COMBINE SIGN BIT WITH HI 7 BITS OF MANTISSA OF RESULT (NOTE: LEADING 1's LOST  5023)
1EFA 77          MOV  M,A PUT BYTE BACK INTO RESULT IN HOLDING AREA                5024
1EFB C9          RET  DONE                                                 I       5025
-------------------------------------------------------------------------------------------------------
1EFC 21 0E 04    X1EFC LXI  H,X040E LOAD HL WITH ADDR OF LO-ORDER BYTE OF 56 BIT MANTISSA  !  5026 ADD 1 TO MANTISSA AT Ø4ØE...Ø414
1EFF 06 07       MVI  B,H'07' SET LOOP CTR TO 7 (7 BYTES IN MEMORY ARE THE MANTISSA)  5027
1F01 34    X1F01 INR  M ADD 1 TO BYTE IN MANTISSA                           4       5028 (56 BIT MANTISSA, EXPONENT OF
                 (DONE)
1F02 C0          RNZ  IF NOT ZERO, DONE    IF ZERO, CARRY 1 INTO NEXT BYTE  @       5029 RESULT SAME AS EXPONENT OF
1F03 23          INX  H ADVANCE PTR TO NEXT BYTE IN MANTISSA                #       5030 LARGER #, IF OVERFLOW, ADD 1
1F04 05          DCR  B DECREMENT LOOP CTR                                         5031 TO EXPONENT, FORCE MANTISSA
1F05 C2 01 1F    JNZ  X1F01 LOOP UNTIL ALL 7 BYTES IN MANTISSA HAVE BEEN INCREMENTED  B  5032 TO 8Ø ØØ... ØØ
1F08 34          INR  M ADJUST EXPONENT BYTE (ADD 1, MULTIPLY # BY 2)      4       5033
                 (ERROR)
1F09 CA 3E 19    JZ   X193E IF ZERO, EXPONENT BYTE OVERFLOW, PRINT ERROR MESSAGE "OVERFLOW"  U>  5034
1F0C 2B          DCX  H BACK-UP PTR TO HI-ORDER BYTE IN MANTISSA (Ø414)    +       5035
1F0D 3E 80       MVI  M,H'80' DIVIDE ENTIRE MANTISSA BY 2 (PUTS OVERFLOW BACK IN MANTISSA)  6  5036
1F0F C9          RET  DONE WITH ROUND-UP ROUTINE                           I       5037
-------------------------------------------------------------------------------------------------------
1F10 11 3B 04    X1F10 LXI  D,X043B LOAD DE WITH ADDR OF LSB OF DIVIDEND   :       5038
1F13 21 1B 04          LXI  H,X041B LOAD HL WITH ADDR OF LSB OF DIVISOR    !       5039 LOAD DE & HL WITH ADDR'S OF 2 #'S
                                                                                    TO BE ADDED OR SUBTRACTED (USED
                                                                                    BY ORI, BASE, DIVIDE)
```

```
1F16  C3 21 1F          JMP   X1F21 JMP INTO TO ADD OR SUBTRACT 7 BYTE MANTISSA'S
1F19  3E 8E      X1F19  MVI   A,H'8E' LOAD ACC WITH OP CODE FOR ADC M
1F1B  21 18 04   X1F1B  LXI   H,X0418 LOAD HL WITH ADDR OF FIRST BYTE OF # IN TEMP STORAGE AREA
1F1E  11 0E 04   X1F1E  LXI   D,X040E LOAD DE WITH ADDR OF FIRST BYTE OF # IN HOLDING AREA
1F21  0E 07      X1F21  MVI   C,H'07' SET LOOP CTR TO 7 (7 BYTES IN #, ADD or SUBTRACT)
1F23  32 28 1F          STA   X1F28 STA CODE FOR TYPE OF OPERATION (9E = SBB, 8E = ADC)
1F26  AF                XRA   A  CLEAR CARRY BIT
1F27  1A         X1F27  LDAX  D FETCH A BYTE FROM # TO CONVERT (OR BYTE OF ARG FOR ADD OR SUB)
1F28  8E         X1F28  ADC   M (ADC M) or (SBB M) - ADD or SUBTRACT A BYTE FROM DECIMAL
1F29  12                STAX  D STORE BYTE BACK INTO HOLDING AREA
1F2A  13                INX   D ADVANCE PTR TO #
1F2B  23                INX   H ADVANCE PTR TO CONSTANT
1F2C  0D                DCR   C DECREMENT LOOP CTR
1F2D  C2 27 1F          JNZ   X1F27 LOOP UNTIL ALL BYTES ADDED (OR SUBTRACTED)
1F30  C9                RET   DONE

1F31  7E         X1F31  MOV   A,M  } COMPLEMENT SIGN BIT IN MEMORY
1F32  2F                CMA      }
1F33  77                MOV   M,A}
1F34  21 0D 04          LXI   H,X040D LOAD HL WITH ADDR OF LOST-BIT BYTE
1F37  06 08             MVI   B,H'08' SET BYTE-LOOP CTR TO 8
1F39  AF                XRA   A     CLEAR ACC
1F3A  4F                MOV   C,A   SAVE ØØ IN REG C
1F3B  79         X1F3B  MOV   A,C  } COMPUTE (M) = ØØ - (M) - CY
1F3C  9E                SBB   M   }
1F3D  77                MOV   M,A }
1F3E  23                INX   H     ADVANCE PTR TO MANTISSA
1F3F  05                DCR   B     DECREMENT LOOP CTR
1F40  C2 3B 1F          JNZ   X1F3B LOOP UNTIL ALL 8 BYTES HAVE BEEN SUBTRACTED
1F43  C9                RET   DONE

1F44  71         X1F44  MOV   M,C STORE MSB BYTE BACK INTO # IN HOLDING AREA
1F45  E5                PUSH  H PUT PTR TO MSB OF # TO ADJUST ON STACK
1F46  06 08      X1F46  SUI   H'08' TEST IF EXPONENT DIFFERENCE IS 7 OR LESS
1F48  DA 5B 1F          JC    X1F5B IF DIFFERENCE IS 7 OR LESS, DO A MULTIPLE BYTE SHIFT RIGHT TO FINISH ADJUSTMENT
1F4B  E1                POP   H RESET HL TO POINT TO MSB OF # TO ADJUST } ALLOWS ALTERNATE ENTRY TO LOOP
1F4C  E5         X1F4C  PUSH  H SAVE PTR TO # BEING ADJUSTED ON STACK
1F4D  11 00 08          LXI   D,X0800 LOAD D WITH 1 more than # of bytes involved in shift operation, load E with ØØ (put in hi byte - same as shift Ø into hi-byte 8 times)
1F50  4E         X1F50  1 MOV  C,M FETCH BYTE FROM # INTO REG C
1F51  73                2 MOV  M,E PUT BYTE IN E INTO # IN MEMORY
1F52  59                3 MOV  E,C PUT BYTE FROM # INTO REG E (EFFECTIVELY SHIFT RIGHT 8 PLACES)
1F53  2B                4 DCX  H DECR PTR TO # IN MEMORY
1F54  15                  DCR  D DECR BYTE CTR
1F55  C2 50 1F            JNZ  X1F50 IF BYTE CTR NOT ØØ, MOVE BYTES IN # TO BE RIGHT SHIFTED
1F58  C3 46 1F            JMP  X1F46 REPEAT EXPONENT DIFFERENCE TEST TIL DIFFERENCE IS LESS THAN 8
1F5B  C6 09      X1F5B  ADI   H'09' ADJUST EXPONENT DIFFERENCE TO POSITIVE #(1 LARGER SINCE LOOP DECR'S CTR FIRST)
1F5D  57                MOV   D,A PUT # TO COUNT LOOPS IN REG D
1F5E  AF         X1F5E  XRA   A CLEAR CARRY BIT (FOR ROTATE RIGHT - 1F66)
1F5F  F1                POP   H RESET HL TO POINT TO MSB OF # TO ADJUST
1F60  15                DCR   D DECREMENT CTR (COUNTS # OF TIMES TO SHIFT # IN MEMORY RIGHT)
1F61  C8         DONE   RZ    IF CTR IS ZERO, ADJUSTMENT IS FINISHED
1F62  E5         X1F62  PUSH  H SAVE PTR TO MSB OF # TO BE ADJUSTED ON STACK
1F63  1E 08             MVI   E,H'08' LOAD D WITH # OF BYTES INVOLVED IN SHIFT OPERATION, Ø417 WILL CONTAIN ACCUMULATED LOST BITS
1F65  7E         X1F65  MOV   A,M FETCH A BYTE FROM # INTO ACC
1F66  1F                RAR     SHIFT RIGHT, CARRY INTO HI-BIT, LO-BIT INTO CARRY
1F67  77                MOV   M,A PUT RESULT BACK INTO # IN MEMORY
1F68  2B                DCX   H DECR PTR TO # (POINTS TO NXT LOWER-ORDER BYTE)
1F69  1D                DCR   E DECR BYTE CTR
1F6A  C2 65 1F          JNZ   X1F65 REPEAT TIL ALL 8 BYTES HAVE BEEN SHIFTED
1F6D  C3 5E 1F          JMP   X1F5E REPEAT LOOP TIL ENTIRE # ADJUSTED

1F70  21 14 04   X1F70  LXI   H,X0414 LOAD HL WITH ADDR OF HI-ORDER BYTE OF MANTISSA OF RESULT
1F73  16 01             MVI   D,H'01' SET SHIFT-RIGHT CTR TO 1 (DIVIDE MANTISSA BY 2 SINCE ADDING 1 TO EXPONENT SAME AS MULTIPLY BY 2)
```

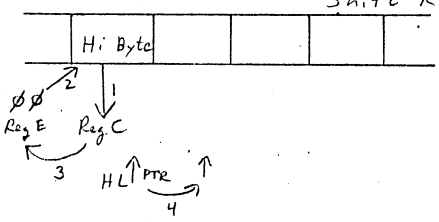
ADD OR SUBTRACT 2 SEVEN-BYTE #'S IN MEMORY

LARGER # } FROM IEAL CONSTANT SMALLER #

COMPUTE Ø - MANTISSA

INVERT SIGN BIT

Shift # in Ø40E - Ø41H Right n places
in Ø413 - Ø41E



```
1F75  C3 62 1F          JMP   X1F62 JMP INTO SHIFT-RIGHT ROUTINE
1F78  0E 08       X1F78 MVI   C,H'08' SET BYTE-LOOP CTR TO 08
1F7A  7E          X1F7A MOV   A,M FETCH A BYTE FROM MANTISSA
1F7B  17                RAL       SHIFT BYTE LEFT 1 BIT   (CARRY USED TO LINK BYTES)
1F7C  77                MOV   M,A PUT BYTE BACK INTO MANTISSA
1F7D  23                INX   H   ADVANCE PTR TO MANTISSA
1F7E  0C                DCR   C   DECREMENT LOOP CTR
1F7F  C2 7A 1F          JNZ   X1F7A LOOP UNTIL 8 BYTES HAVE BEEN SHIFTED
1F82  C9                RET       DONE
1F83  EF                RST   5   PERFORM FLOATING-PT SIGN TEST ON # IN HOLDING AREA
1F84  C8                RZ        IF # IS ZERO, THEN MULTIPLY IS DONE, RESULT IS ALREADY IN HOLDING AREA
1F85  CC A1 1A          CALL  X1AA1 COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS
1F88  CC 18 20          CALL  X2018 MOVE MULTIPLIER  FROM 04ØE-0414 TO 043B-0441, CLEAR HOLDING AREA
1F8B  71                MOV   M,C CLEAR LOC 040D (LOST-BIT BYTE OF ACCUMULATED PRODUCT)
1F8C  13                INX   D ADVANCE PTR TO MULTIPLIER  TO LOC 043B (LOWEST-ORDER BYTE OF MANTISSA)
1F8D  06 07             MVI   B,H'07' SET BYTE-LOOP CTR TO 7 (7 BYTES OF MANTISSA TO MULTIPLY)
1F8F  1A          X1F8F LDAX  D FETCH BYTE FROM MULTIPLIER INTO ACC
1F90  13                INX   D ADVANCE PTR TO NEXT BYTE IN MULTIPLIER
1F91  B7                ORA   A TEST MULTIPLIER BYTE
1F92  D5                PUSH  D SAVE PTR TO MULTIPLIER ON STACK
1F93  CA AF 1F          JZ    X1FAF IF MULTIPLIER BYTE IS ZERO, SHIFT ACCUMULATED PRODUCT 8 BITS RIGHT, SINCE NOTHING TO ADD TO ACCUMULATED PRODUCT
1F96  0E 08             MVI   C,H'08' SET BIT-LOOP CTR TO 8 (8 BITS OF MULTIPLIER TO TEST)
1F98  C5          X1F98 PUSH  B SAVE BYTE-LOOP (see 1F8D) & BIT-LOOP (see 1F96) CTRS ON STACK
1F99  1F                RAR   ROTATE LO-BIT OF MULTIPLIER INTO THE CARRY BIT
1F9A  47                MOV   B,A SAVE THE REST OF THE MULTIPLIER BYTE IN REG B
1F9B  DC 19 1F          CC    X1F19 IF MULTIPLIER BIT WAS A 1, ADD MULTIPLICAND (0418-041E) TO ACCUMULATED PRODUCT (04ØE-0414)-7 BYTE MANTISSA
1F9E  CD 70 1F          CALL  X1F70 SHIFT ACCUMULATED PRODUCT & LOST BIT BYTE RIGHT 1 BIT
1FA1  78                MOV   A,B PUT REST OF MULTIPLIER IN ACC
1FA2  C1                POP   B FETCH BYTE-LOOP & BIT-LOOP CTRS FROM STACK
1FA3  0D                DCR   C DECREMENT BIT-LOOP CTR
1FA4  C2 98 1F          JNZ   X1F98 LOOP UNTIL ALL 8 BITS OF MULTIPLIER HAVE BEEN USED
1FA7  D1          X1FA7 POP   D FETCH PTR TO MULTIPLIER FROM STACK
1FA8  05                DCR   B DECREMENT BYTE LOOP CTR
1FA9  C2 8F 1F          JNZ   X1F8F LOOP UNTIL ALL 7 BYTES OF MULTIPLIER HAVE BEEN USED
1FAC  C3 80 1E          JMP   X1E80 JMP TO ROUTINE TO NORMALIZE PRODUCT, RESTORE SIGN BIT OF RESULT,
1FAF  21 14 04    X1FAF LXI   H,X0414 LOAD HL WITH ADDR OF MSB OF ACCUMULATED PRODUCT IN HOLDING AREA
1FB2  CD 4C 1F          CALL  X1F4C SHIFT ACCUMULATED PRODUCT & LOST-BIT BYTE RIGHT 8 PLACES
1FB5  C3 A7 1F          JMP   X1FA7 JMP BACK INTO LOOP
1FB8  00          X1FB8 NOP
1FB9  00                NOP
1FBA  00                NOP
1FBB  00                NOP
1FBC  00          X1FBC NOP
1FBD  00                NOP
1FBE  20                DATA  A' '
1FBF  84                ADD   H
1FC0  11 B8 1F    X1FC0 LXI   D,X1FB8 LOAD DE WITH ADDR OF DBL-PREC CONSTANT 10 DECIMAL
1FC3  21 18 04          LXI   H,X0418 LOAD HL WITH ADDR OF DBL-PREC TEMP STORAGE AREA
1FC6  CD 66 13          CALL  X1366 MOVE CONSTANT INTO TEMP STORAGE AREA
1FC9  3A 1F 04          LDA   X041F
1FCC  B7                ORA   A
1FCD  CA AC 04          JZ    X04AC
1FD0  CD 9E 1A          CALL  X1A9E COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS
1FD3  34                INR   M
1FD4  34                INR   M
1FD5  CD 18 20          CALL  X2018 MOVE DIVIDEND FROM 04ØE-0414 TO 043B-0441, CLEAR HOLDING AREA - USED TO
1FD8  21 42 04          LXI   H,X0442 LOAD HL WITH ADDR OF EXPONENT BYTE OF DIVIDEND
1FDB  71                MOV   M,C CLEAR LOC 0442 (OVERFLOW BYTE OF DIVIDEND)
1FDC  41                MOV   B,C CLEAR REG B - USED TO TELL IF QUOTIENT IS ØØ -
1FDD  3E 9E       X1FDD MVI   A,H'9E' LOAD ACC WITH OP CODE FOR SBB M - see 2ØØ9-2ØØB
```

SHIFT 56 BIT MANTISSA + LOST-BIT BYTE LEFT 1 BIT

DOUBLE-PRECISION MULTIPLY (# IN 04ØE-0415) X (# IN 0418-041F) TO ACCUMULATE PRODUCT

DOUBLE-PRECISION FLOATING-POINT CONSTANT = 10 DECIMAL

SINGLE-PRECISION FLOATING-POINT CONSTANT = 10 DECIMAL

TEST DIVISOR STORED IN TEMP STORAGE AREA 0418.... IF DIVISOR IS ZERO, PRINT ERROR MESSAGE "DIVISION BY ZERO"

DOUBLE PRECISION DIVIDE (# IN 04ØE-0415) ÷ (# IN 0418-041F) ACCUMULATE QUOTIENT

```
                                                  (Ø43B-Ø441) (Ø438-Ø441)  (Ø418-Ø41E)
1FDF  CC 10 1F          CALL X1F10 COMPUTE DIVIDEND = DIVIDEND - DIVISOR                M   5160
1FE2  1A                LDAX D LOAD ACC WITH OVERFLOW BYTE OF DIVIDEND (LOC Ø442)        5161
1FE3  99                SBB  C SUBTRACT CARRY BIT (OVERFLOW FROM HI BYTE OF DIVIDEND)   5162
1FE4  3F                CMC  COMPLEMENT CARRY BIT (FOR CONVENIENCE IN SETTING UP THE JMP-ON-CONDITION) 5163
1FE5  DA EF 1F          JC   X1FEF JMP IF SUBTRACT IS LEGITIMATE                        Z   5164
1FE8  3E 8E             MVI  A,H'8E' LOAD ACC WITH OP CODE FOR ADC M                    >   5165
1FEA  CC 10 1F          CALL X1F10 ADD DIVISOR TO DIVIDEND - PREVIOUS SUBTRACT NOT LEGITIMATE M 5166
1FED  AF                XRA  A CLEAR CARRY BIT (FORCES SKIP OF NEXT INSTRUCTION)        /   5167
1FEE  DA 12 04    STAX D INR B JC   X0412 Dummy IF SUBTRACT WAS LEGITIMATE, STORE OVERFLOW BYTE BACK INTO MEMORY 5168 INCREMENT B TO SHOW QUOTIENT IS NOT ØØ
1FF1  3A 14 04          LDA  X0414 } TEST MSB OF QUOTIENT, IF MSB IS A 1, DIVISION IS COMPLETED TO THE 5169
1FF4  3C                INR  A     } MAXIMUM PRECISION (56 BITS OF QUOTIENT)            <   5170
1FF5  3C                DCR  A     }                                                    =   5171
1FF6  1F                RAR  PUT LAST BIT (CARRY OR NO CARRY FROM SUBTRACT) IN MSB OF ACC FOR ROUND OFF TEST AT 1EED - IEEE 5172
1FF7  FA ED 1E          JM   X1EED IF MSB OF QUOTIENT IS A 1, DIVISION LOOP COMPLETE, PERFORM ROUNDOFF, NORMALIZATION, AND PUT SIGN BIT INTO RESULT 5173
1FFA  17                RAL  PUT LAST BIT BACK INTO CARRY BIT TO SHIFT IT INTO THE 7 QUOTIENT BYTES 5174
1FFB  21 0E 04          LXI  H,X040E LOAD HL WITH ADDR OF LSB OF QUOTIENT               !   5175
1FFE  0E 07             MVI  C,H'07' SET BYTE-LOOP CTR TO 7 (7 BYTES OF MANTISSA TO SHIFT) 5176
2000  CD 7A 1F          CALL X1F7A SHIFT CARRY BIT INTO 7 BYTES OF QUOTIENT, IF SUBTRACT OK, CARRY IS SET M , 5177 OTHERWISE SHIFT A Ø INTO QUOTIENT
2003  21 3B 04          LXI  H,X043B LOAD HL WITH LSB OF DIVIDEND (WHAT'S LEFT OF IT)   !;  5178
2006  CD 78 1F          CALL X1F78 SHIFT WHAT'S LEFT OF THE DIVIDEND LEFT 1 BIT (SAME AS SHIFTING THE 5179 DIVISOR RIGHT 1 PLACE) - INCLUDES OVERFLOW BYTE
2009  78                MOV  A,B } TEST REG B - IF ENTIRE QUOTIENT IS ZERO, ADJUST EXPONENT BY 1      5180
200A  B7                ORA  A   } (see 1A8C - 1A96 - SNG-PREC DIVIDE)                  7   5181
200B  C2 00 1F          JNZ  X1F00 JMP IF QUOTIENT IS NOT ØØ                            8}  5182
200E  21 15 04          LXI  H,X0415 } DECREMENT EXPONENT RESULT BY 1                   !   5183
2011  35                DCR  M       }                                                  5   5184
2012  C2 00 1F          JNZ  X1F00 IF EXPONENT BYTE IS NOT ZERO, JMP TO 1F00 TO CONTINUE DIVISION 8} 5185
2015  C3 3E 19          JMP  X193E IF EXPONENT BYTE IS ZERO, PRINT ERROR MESSAGE "OVERFLOW"  C>  5186
------------------------------------------------------------------------------------------------------
2018  79          X2018 MOV  A,C  } PUT MSB OF 2ND # WITH RESTORED LEADING 1 BACK INTO # IN MEMORY 5187
2019  32 1E 04          STA  X041E }                                                    2   5188
201C  2B                DCX  H DECREMENT PTR TO HOLDING AREA (POINT TO LOC Ø414)         +   5189
201D  11 41 04          LXI  D,X0441 LOAD DE WITH ADDR OF MSB BYTE OF TEMP STORAGE TO HOLD MULTIPLICAND 5190 OR DIVIDEND
2020  01 00 07          LXI  B,X0700 SET BYTE-LOOP CTR (REG B) TO 7, SET REG C TO ØØ - USED TO CLEAR 5191 HOLDING AREA - USED FOR PRODUCT OR QUOTIENT
2023  7E          X2023 MOV  A,M FETCH A BYTE FROM 1ST # (IN HOLDING AREA)               5192
2024  12                STAX D  STORE BYTE IN A TEMP STORAGE AREA                        5193
2025  71                MOV  M,C CLEAR THE BYTE IN HOLDING AREA                         5194
2026  1B                DCX  D  DECREMENT DESTINATION PTR                               5195
2027  2B                DCX  H  DECREMENT SOURCE PTR                                    +   5196
2028  05                DCR  B  DECREMENT BYTE-LOOP CTR                                 5197
2029  C2 23 20          JNZ  X2023 LOOP UNTIL ALL 7 BYTES HAVE BEEN MOVED               BØ  5198
202C  C9                RET  DONE                                                       I   5199
------------------------------------------------------------------------------------------------------
202D  CD 91 18    X202D CALL X1891 COPY DBL-PREC # AT Ø43E... TO TEMP STORAGE AT Ø418    M   5200
2030  EB                XCHG PUT PTR TO ACCUMULATED VALUE IN HL, PTR TO # IN TEMP STORAGE IN DE 5201 MULTIPLY DBL-PREC # IN
2031  2B                DCX  H } PUT EXPONENT BYTE OF ACCUMULATED VALUE IN ACC          +   5202 HOLDING AREA BY 10 DECIMAL
2032  7E                MOV  A,M } (1891 PUSHES HL 1 ADDR TOO FAR - REASON FOR DCX)         5203
2033  C6 02             ADI  H'02' ADD 2 TO EXPONENT (MULTIPLY DBL-PREC # BY 4)         F   5204
2035  DA 3E 19          JC   X193E IF CARRY OCCURS, EXPONENT TOO BIG, PRINT ERROR MESSAGE Z>  5205
2038  77                MOV  M,A PUT EXPONENT BYTE BACK INTO #  "OVERFLOW"               5206
2039  E5                PUSH H SAVE ADDR ON STACK                                       5207
203A  CD 4A 1E          CALL X1E4A ADD THE 2 #'S TOGETHER (FORM 5X # IN HOLDING AREA)   MJ  5208
203D  E1                POP  H FETCH ADDR OF EXPONENT BYTE OF # IN HOLDING AREA INTO HL 5209
203E  34                INR  M MULTIPLY # IN HOLDING AREA BY 2 (FORMS 10X # IN HOLDING AREA) 4) 5210
203F  C0                RNZ  IF EXPONENT BYTE NOT ZERO, MULTIPLY IS OK AND FINISHED     2   5211
2040  C3 3E 19          JMP  X193E IF CARRY OCCURS, EXPONENT TOO BIG, PRINT ERROR MESSAGE "OVERFLOW" 5212
------------------------------------------------------------------------------------------------------
2043  FE 2D       X2043 CPI  A'-' TEST IF 1ST CHAR IS A MINUS SIGN                      -   5213
2045  F5                PUSH PSW  PUT CHAR & FLAGS ON STACK                             5214 CONVERT STRING TO NUMERIC
2046  CA 4F 20          JZ   X204F JMP IF CHAR IS '-'                                   JO  5215 VALUE,
2049  FE 2B             CPI  A'+' TEST IF CHAR IS A PLUS SIGN                           +   5216
204B  CA 4F 20          JZ   X204F JMP IF CHAR IS '+'                                   JO  5217
204E  2B                DCX  H IF CHAR IS NOT A SIGN CHAR, BACK-UP THE SCAN POINTER TO 1ST 5218
204F  EB          X204F XCHG PUT PTR TO CHAR STRING TO CONVERT IN DE                     CHAR 5219
```

```
2050  01 F. J0          LXI  B,X00FF PUT -1 IN C, PUT ØØ IN B                    5220
2053  60                MOV  H,B } INITIALIZE HL TO ØØØØ - ACCUMULATED VALUE STARTS AS   5221
2054  68                MOV  L,B }                    AN INTEGER VALUE ØØØØ      5222
2055  CD 20 1C          CALL X1C20 INITIALIZE VALUE HOLDING AREA TO ØØØØ + INTEGER TYPE M-  5223
2058  E9                XCHG PUT PTR TO CHAR STRING IN HL, PUT ØØØØ IN DE          5224
2059  D7          X2059 RST  2   SCAN TO NEXT CHAR (SIGN CHAR, IF ANY, IS ON STACK - see 2045) W  5225
205A  DA E8 20    DIGIT JC   X20E8 JMP IF CHAR IS A DIGIT -                       Z  5226
205D  FE 2E             CPI  A'.' } TEST CHAR, JMP IF '.'                        .  5227
205F  CA AF 20     '.'  JZ   X20AF }                                             J/ 5228
2062  FE 45             CPI  A'E' } TEST CHAR, JMP IF 'E', CONVERT VALUE IN HOLDING AREA TO SNG-PREC, CONTINUE TO PROCESS EXPONENT 5229
2064  CA 60 20    'E'   JZ   X2060 }                                            J  5230
2067  FE 44             CPI  A'D' } TEST CHAR, JMP IF NOT 'D'                    D  5231
2069  C2 91 20  NOT 'D' JNZ  X2091 }                                            B  5232
206C  B7          'D'   ORA  A  RESET THE ZERO FLAG (CHAR IN ACC IS 'D')        7  5233
206D  CD BA 20    X206D CALL X20BA CONVERT # IN HOLDING AREA TO SNG-PREC IF ZERO FLAG SET, CONVERT TO DBL PREC IF ZERO FLAG RESET H3  5234
2070  E5                PUSH H       } PUT ADDR 2Ø86 ON STACK FOR IMPLIED JMP      5235
2071  21 86 20          LXI  H,X2086 }                                             I  523E
2074  E3                XTHL         } LEAVE PTR TO CHAR STRING IN HL              5237
2075  D7                RST  2  SCAN TO NEXT NON-SPACE CHAR IN STRING              H  5238
2076  15                DCR  D  DECREMENT D TO -1                                  5239
2077  FE AF             CPI  H'AF' } JMP TO 2Ø86 WITH D = -1 IF CHAR IS MINUS SIGN OR  /  5240
2079  C8          '-'EXP RZ       } TOKEN FOR A MINUS SIGN                         H  5241
207A  FE 2D             CPI  A'-'                                                  -  5242
207C  C8          '-'EXP RZ                                                        H  5243
207D  14                INR  D  INCREMENT D TO ØØ                                  5244
207E  FE AE             CPI  H'AE' } JMP TO 2Ø86 WITH D = ØØ IF CHAR IS PLUS SIGN OR  .  5245
2080  C8          '+'EXP RZ       } TOKEN FOR A PLUS SIGN                          H  5246
2081  FE 2B             CPI  A'+'                                                  +  5247
2083  C8          '+'EXP RZ                                                        H  5248
2084  2B                DCX  H  IF CHAR IS NOT A SIGN CHAR OR SIGN TOKEN, BACK-UP SCAN PTR  +  5249
2085  F1                POP  PSW REMOVE IMPLIED JMP ADDR 2Ø86 FROM STACK           5250
2086  D7          X2086 RST  2  SCAN TO NEXT NON-SPACE CHAR                        W  5251
2087  DA 57 21  DIGIT   JC   X2157 JMP IF CHAR IS A DIGIT, IF NOT A DIGIT, THEN IT MUST BE A DELIMITER  ZW!  5252
208A  14                INR  D  INCREMENT SIGN OF EXPONENT BYTE (SETS THE FLAGS CONVENIENTLY)  5253
208B  C2 91 20          JNZ  X2091 IF EXPONENT IS POSITIVE, JMP TO 2Ø91            B  5254
208E  AF                XRA  A  CLEAR ACC          } FORM 2'S COMPLEMENT OF        /  5255
208F  93                SUB  E  SUBTRACT EXPONENT MAGNITUDE } MAGNITUDE (SINCE MINUS EXPONENT)  5256
2090  5F                MOV  E,A PUT # BACK INTO REG E                             -  5257
2091  E5          X2091 PUSH H  SAVE PTR TO CHAR STRING ON STACK                   5258
2092  78                MOV  A,E } SUBTRACT #-OF-DIGITS-ON-RIGHT-OF-DECIMAL-PT - FROM EXPONENT MAGNITUDE  E  5259
2093  90                SUB  B   } (MANTISSA IS AN INTEGER AT THIS POINT           5260
2094  F4 C9 20    X2094 CP   X20C9 IF EXPONENT IS POSITIVE, MULTIPLY MANTISSA BY 1Ø, DECREMENT ACC BY 1  I  5261
2097  FC D7 20          CM   X20D7 IF EXPONENT IS NEGATIVE, DIVIDE MANTISSA BY 1Ø, INCREMENT ACC BY 1  W  5262
209A  C2 94 20          JNZ  X2094 LOOP UNTIL EXPONENT MAGNITUDE REDUCED TO Ø      B  5263
209D  E1                POP  H  FETCH PTR TO CHAR STRING FROM STACK                5264
209E  F1                POP  PSW FETCH SIGN CHAR & FLAGS FROM STACK                5265
209F  E5                PUSH H  SAVE PTR TO CHAR STRING ON STACK                   5266
20A0  CC 0C 1B          CZ   X1B0C ZERO FLAG SET IF ENTIRE # IS MINUS (see 2Ø43), NEGATE VALUE IN HOLDING AREA  5267
20A3  E1                POP  H  FETCH PTR TO CHAR STRING FROM STACK                5268
20A4  F7                RST  6  TEST FOR VALUE TYPE IN HOLDING AREA                5269
20A5  E8          DONE  RPE  CONVERSION FINISHED IF INTEGER OR DBL-PREC VALUE      5270
20A6  E5                PUSH H  SAVE PTR TO CHAR STRING ON STACK                   5271  SPECIAL CASE: -32768 CAN BE
20A7  21 1C 1A          LXI  H,X1A1C } PUT PSEUDO ERROR MESSAGE ADDR ON STACK, IF SNG-PREC # IS NOT  5272  STORED AS AN INTEGER -
20AA  E5                PUSH H       } THE SPECIAL CASE, RET TO IAIC WHICH POPS H (PTR TO STRING IN HL)  5273  FORMAT VALUE. TEST IF
20AB  CD 36 1C          CALL X1C36 IF SNG-PREC VALUE IS SPECIAL CASE, CONVERT TO INTEGER FORMAT  M6  5274  SNG-PREC SHOULD BE AN INTEGER
20AE  C9          DONE  RET  PUT PTR TO CHAR STRING BACK IN HL BEFORE EXITING THIS ROUTINE  I  5275
                                                                                 5276
20AF  F7          X20AF RST  6  TEST VALUE TYPE OF ACCUMULATED VALUE IN HOLDING AREA  5277  DECIMAL-PT FOUND IN
20B0  0C                INR  C  CHANGE C TO ØØ - INDICATES NEXT GROUP OF DIGITS HAVE FRACTIONAL VALUE  5278  MANTISSA
20B1  C2 91 20          JNZ  X2091 IF C IS NOT Ø, SECOND DECIMAL-PT FOUND - END OF MAGNITUDE EVALUATION  B  5279
20B4  DC 9A 20          CC   X209A CONVERT ACCUMULATED VALUE TO SINGLE PRECISION
20B7  C3 59 20          JMP  2059 CONTINUE PROCESSING NEXT CHAR
```

```
2087  C3 F 20          JMP  X2059                                          CY   5280
                                                                              5281
208A  E5          X208A PUSH H  }                                          U    5282 CONDITIONAL CONVERSION TO
208B  D5                PUSH D  } SAVE ALL REGISTERS ON THE STACK           E    5283 SNG- OR DBL-PRECISION (# IN
208C  C5                PUSH B  }                                               5284 DBL PREC)   HOLDING AREA)
208D  F5                PUSH PSW SAVE FLAGS ON STACK (ZERO FLAG SET- SNG PREC, ZERO FLAG RESET-
208E  CC 45 1C          CZ   X1C45 CONVERT VALUE IN HOLDING AREA TO SINGLE-PRECISION, IF ZERO FLAG  LE   5285 SET
20C1  F1                POP  PSW FETCH FLAGS FROM STACK                         5286
20C2  C4 6F 1C          CNZ  X1C6F CONVERT VALUE IN HOLDING AREA TO DOUBLE-PRECISION, IF ZERO  O       5287
20C5  C1                POP  B  } RESTORE REGISTERS TO ORIGINAL VALUES FROM STACK  FLAG  A   5288 RESET
20C6  D1                POP  D  }                                           Q    5289
20C7  E1                POP  H  }                                               5290
20C8  C9                RET  END OF CONDITIONAL CONVERSION                  I    5291
                                                                            H    5292
20C9  C8          X20C9 RZ   RETURN IF EXPONENT CTR IS Ø (SINCE THIS ROUTINE IS CALLED BY A POSITIVE CONDITION  5293 (see 2094
20CA  F5          X20CA PUSH PSW SAVE EXPONENT CTR ON STACK                  5294
20CB  F7                RST  6 TEST VALUE TYPE OF ACCUMULATED VALUE STORED IN HOLDING AREA
20CC  F5                PUSH PSW SAVE FLAG STATUS ON STACK                  5295 } MULTIPLY ACCUMULATED VALUE
20CD  E4 04 1A          CPO  X1A04 MULTIPLY SINGLE-PREC # IN HOLDING AREA BY 10 DECIMAL  T  5296 } BY 10 DECIMAL
2000  F1                POP  PSW FETCH FLAG STATUS FROM STACK               5297 }
2001  EC 20 20          CPE  X2020 MULTIPLY DOUBLE-PREC # IN HOLDING AREA BY 10 DECIMAL  -  5298 }
2004  F1                POP  PSW FETCH EXPONENT CTR FROM STACK                  5299
2005  3C          X2005 DCR  A DECREMENT EXPONENT CTR (EACH POSITIVE 1 IN EXPONENT SAME AS=  5300 MULTIPLY BY 10
2006  C9                RET  DONE                                           I    5301
                                                                            U    5302
2007  D5          X2007 PUSH D  } SAVE REGISTERS ON STACK                       5303
2008  E5                PUSH H  }                                               5304
2009  F5                PUSH PSW SAVE EXPONENT CTR ON STACK                     5305
200A  F7                RST  6 TEST VALUE TYPE OF ACCUMULATED VALUE STORED IN HOLDING AREA
200B  F5                PUSH PSW SAVE FLAG STATUS ON STACK                  5306 } MULTIPLY ACCUMULATED VALUE
200C  E4 23 1A          CPO  X1A23 DIVIDE SINGLE-PRECISION # IN HOLDING AREA BY 10 DECIMAL  #  5307 } BY .1 DECIMAL
200F  F1                POP  PSW FETCH FLAG STATUS FROM STACK               5308 }
20E0  EC C0 1F          CPE  X1FC0 DIVIDE DOUBLE-PRECISION # IN HOLDING AREA BY 10 DECIMAL  @  5309 }
20E3  F1                POP  PSW                                                5310
20E4  E1                POP  H  } RESTORE REGISTERS TO ORIGINAL VALUES FROM STACK  5311
20E5  D1                POP  D  }                                           Q    5312
20E6  3C                INR  A INCREMENT EXPONENT CTR (EACH NEGATIVE 1 IN EXPONENT SAME AS<  5313 DIVIDE BY 10)
20E7  C9                RET  DONE                                           I    5314
                                                                            U    5315 Ø HERE)
20E8  D5          X20E8 PUSH D SAVE EXPONENT SIGN (D) + EXPONENT MAGNITUDE (E) ON STACK (BOTH ARE
20E9  78                MOV  A,B } ENTRY TO 20E8 WITH CARRY SET (see 20F5H), B IS Ø, C IS -1 ORIGINALLY,  5316 WHEN DECIMAL PT IS FOUND, C BECOMES ØØ
20EA  89                ADC  C   } EACH DIGIT ON RIGHT OF DECIMAL-PT ADDS 1 TO B.                          5317
20EB  47                MOV  B,A }                                          G    5318
20EC  C5                PUSH B  SAVE B (# OF DIGITS TO RIGHT OF DECIMAL-PT) + C (DECIMAL-PT INDICATOR) E ON  5319 STACK
20ED  E5                PUSH H  SAVE PTR TO CHAR STRING ON STACK                5320
20EE  7E                MOV  A,M FETCH CHAR FROM STRING INTO ACC                5321
20EF  D6 30             SUI  A'0' } CONVERT ASCII CHAR TO A DIGIT-WEIGHT (INTEGER FORMAT)  VO  5322
20F1  F5                PUSH PSW } SAVE DIGIT-WEIGHT ON STACK                   5323
20F2  F7                RST  6  TEST VALUE TYPE OF # IN HOLDING AREA            5324
20F3  F2 1D 21    FLT'G PT JP  X211D JMP IF FLOATING-PT TYPE OF VALUE (SNG- OR DBL-PREC)  !  5325
20F6  2A 12 04          LHLD X0412 LOAD HL WITH ACCUMULATED INTEGER VALUE FROM HOLDING AREA  *  5326
20F9  11 CD 0C          LXI  D,XØCCD LOAD DE WITH 3277 DECIMAL (MAX POSITIVE INTEGER # IS 32,767.  5327
20FC  E7                RST  4 TEST IF ACCUMULATED INTEGER VALUE IS TOO LARGE TO ACCOMODATE  5328 ANOTHER DIGIT (VALUE * 10 + DIGIT-WEIGHT)
20FD  D2 19 21  CONVERT TO SNG-PREC JNC  X2119 IF HL ≥ DE, ACCUMULATED VALUE TOO LARGE FOR INTEGER FORMAT  !  5329 CONVERT TO SINGLE-PRECISION - JMP
                                                                            T    5330
2100  54          X2100 MOV  D,H } STORE # IN HL IN DE } MULTIPLY ACCUMULATED VALUE  )  5331
2101  50                MOV  E,L }                      } IN HL BY 10 DECIMAL    )  5332
2102  29                DAD  H 2X                       }                        )  5333
2103  29                DAD  H 4X                       }                        .  5334
2104  19                DAD  D +1=5X                    }                        )  5335
2105  29                DAD  H X2=10X                   }                           5336
2106  F1                POP  PSW FETCH DIGIT-WEIGHT FROM STACK                  0  5337
2107  4F                MOV  C,A } ADD DIGIT-WEIGHT TO ACCUMULATED VALUE        5338
2108  09                DAD  B   }                                              5339
2109  7C                MOV  A,H PUT HI-BYTE OF ACCUMULATED VALUE IN ACC
```

```
210A  B7                  ORA  A TEST HI-BYTE,                                                    7     5340
210B  FA 17 21            JM   X2117 JMP IF D7 SET, ACCUMULATED VALUE TOO LARGE FOR 15 BIT INTEGER 5341-CONVERT ACCUMULATED VALUE TO SNG-PREC
210E  22 12 04            SHLD X0412 STORE UPDATED ACCUMULATED VALUE BACK INTO HOLDING AREA          5342
2111  E1            X2111 POP  H PUT PTR TO CHAR STRING IN HL                                       5343
2112  C1                  POP  B FETCH DECIMAL-PT FLAG & CTR FROM STACK (see 20EC)                  A     5344
2113  D1                  POP  D FETCH EXPONENT SIGN + MAGNITUDE FROM STACK (see 20E8)              Q     5345
2114  C3 59 20  NEXT CHAR JMP  X2059 CONTINUE PROCESSING NEXT CHAR IN STRING                        CY    5346
2117  79            X2117 MOV  A,C } PUT DIGIT-WEIGHT ON STACK, CONVERT ACCUMULATED VALUE TO       5347
2118  F5                  PUSH PSW } FLOATING-PT, THEN ADD THE DIGIT WEIGHT                         5348
2119  CD 60 1C      X2119 CALL X1C60 CONVERT INTEGER IN HL TO SINGLE-PRECISION FORMAT              M     5349
211C  37                  STC  FORCE CARRY BIT SET, SINCE VALUE IN HOLDING JUST INCREASED IN SIZE TO 7 SNG- 5350 PRECISION FORMAT
211D  D2 39 21      X211D JNC  X2139 JMP IF DBL-PRECISION VALUE TYPE                              R9!   5351
2120  01 74 94            LXI  B,X9474 } LOAD BCDE WITH 4-BYTE FLOATING-PT # 1E6                      5352
2123  11 00 24            LXI  D,X2400 } B IS EXPONENT BYTE                                   $        5353
2126  CD A1 1B            CALL X1BA1 } TEST IF # IN HOLDING AREA IS 1E6 OR LARGER, IF IT IS, CONVERT M!  5354 ACCUMULATED VALUE TO DBL-PRECISION
2129  F2 36 21            JP   X2136 } (USES FLOATING-PT SUBTRACT SIGN TEST ROUTINE)                  6!    5355
212C  CD D4 1A            CALL X1AD4 MULTIPLY FLOATING-PT # (ACCUMULATED #) BY 10 DECIMAL       MT    5356
212F  F1                  POP  PSW FETCH DIGIT-WEIGHT FROM STACK                                       5357
2130  CD 4C 21            CALL X214C CONVERT DIGIT-WEIGHT TO FLOATING-PT, ADD TO ACCUMULATED #  ML!   5358
2133  C3 11 21  NXT CHAR  JMP  X2111 RESTORE PTRS FROM STACK, CONTINUE PROCESSING NEXT CHAR IN  C !   5359 STRING
2136  CD 77 1C      X2136 CALL X1C77 CONVERT ACCUMULATED VALUE IN HOLDING AREA TO DBL-PREC        M     5360
2139  CD 20 20      X2139 CALL X2020 MULTIPY ACCUMULATED DBL-PREC # BY 10 DECIMAL                 M-    5361
213C  CD 91 1B            CALL X1B91 MOVE ACCUMULATED DBL-PREC # FROM HOLDING AREA TO TEMP STORAGE    5362 AT 0418
213F  F1                  POP  PSW FETCH DIGIT-WEIGHT FROM STACK                                       5363
2140  CD F5 1A            CALL X1AF5 CONVERT DIGIT-WEIGHT TO SNG-PREC # IN HOLDING AREA         M     5364
2143  CD 77 1C            CALL X1C77 CONVERT DIGIT-WEIGHT TO DBL-PREC # IN HOLDING AREA         M     5365
2146  CD 4A 1E            CALL X1E4A USE DBL-PREC ADDITION TO ADD THE 2 #'S TOGETHER            MJ    5366
2149  C3 11 21  NXT CHAR  JMP  X2111 RESTORE PTRS FROM STACK, CONTINUE PROCESSING NEXT CHAR IN  C !   5367 STRING
214C  CD 36 1B      X214C CALL X1B36 PUT SINGLE-PREC # IN HOLDING AREA ON STACK                   M6    5368
214F  CD F5 1A            CALL X1AF5 CONVERT DIGIT-WEIGHT TO SINGLE-PREC #, STORE IN HOLDING AREA M    5369
2152  C1            X2152 POP  B } PUT FLOATING # (ACCUMULATED VALUE) IN BCDE FROM STACK              5370
2153  D1                  POP  D }                                                            Q     5371
2154  C3 9C 18            JMP  X189C USE FLOATING-PT ADDITION ROUTINE TO ADD THE 2 #'S TOGETHER      5372
2157  78            X2157 MOV  A,E  PUT ACCUMULATED EXPONENT VALUE IN ACC                              5373
2158  FE 0A               CPI  H'0A' } IF ACCUMULATED VALUE IS 10 OR GREATER, FORCE EXPONENT TO 50   5374
215A  D2 66 21            JNC  X2166 } (FORCES OVERFLOW                                       R !    5375
215D  07                  RLC  }                                                                     5376
215E  07                  RLC  }  MULTIPLY ACCUMULATED EXPONENT VALUE BY 10, DECIMAL                 5377
215F  83                  ADD  E }                                                                   5378
2160  07                  RLC  }                                                                     5379
2161  86                  ADD  M  } ADD DIGIT-WEIGHT TO EXPONENT VALUE                               5380
2162  D6 30               SUI  A'0' }                                                         VO     5381
2164  5F                  MOV  E,A  PUT EXPONENT VALUE BACK IN REG E                          -      5382
2165  FA 1E 32            JM   X321E DUMMY MVI E,H'32'                                          2    5383
2168  C3 86 20            JMP  X2086 LOOP TO ACCUMULATE MORE EXPONENT DIGITS                  C      5384
216B  E5            X216B PUSH H    SAVE HL ON STACK (LINE #)                                          5385
216C  21 43 04            LXI  H,X0443 LOAD HL WITH ADDR OF CHAR STRING "␣IN␣".               !C     5386 PRINT "␣IN␣(LINE #)"
216F  CD 41 12            CALL X1241 PRINT CHAR STRING                                        MA     5387 LINE # IN HL
2172  E1                  POP  H    RESTORE HL FROM STACK (LINE #)                                   5388
2173  CD 2D 1C      X2173 CALL X1C2D STORE H,L IN 0412+0413, STORE 02 IN 03A4 (HL = LINE #)    M-     5389 STORE # TO CONV IN HOLDING AREA
2176  AF                  XRA  A    CLEAR ACC                                                 /      5390 SET TYPE CODE TO INTEGER TYPE
2177  CD 02 22            CALL X2202 CLEAR 03CD, PUT SPACE AT FIRST LOC IN CHAR BUFR          M "    5391
217A  B6                  ORA  M PUT '␣' IN ACC                                               6      5392          CONVERT & PRINT LINE
217B  CD 9E 21            CALL X219E CONVERT # IN HOLDING AREA (0412 & 0413) TO ASCII IN CHARM !     5393 BUFR  # IN HL
217E  C3 40 12            JMP  X1240 PRINT CHAR STRING (CONVERTED #)                          C@     5394
2181  AF            X2181 XRA  A CLEAR BIT-FLAG BYTE (UNFORMATTED CONVERSION)  see p 88 for BIT-FLAG FORMAT  5395 CONVERT # IN HOLDING AREA TO
2182  CD 02 22      X2182 CALL X2202 STORE BIT-FLAG AT LOC 03CD, PUT SPACE IN 1ST LOC OF CONVERSION BUFFER M "  5396 EQUIVALENT CHAR STRING
2185  E6 08               ANI  H'08' TEST BIT-FLAG BYTE - '+'=1  '-'=0 (see p 88) - 08 BIT IS SIGN BIT    5397
2187  CA 9C 21            JZ   X219C JMP IF !+' (LEADING OR TRAILING, IS NOT TO BE PRINTED)   J !    5398 WHEN DONE HL PTS TO 1ST CHAR OF
218A  36 2B               MVI  M,A'+' PUT A '+' IN CHAR BUFR INSTEAD OF LEADING SPACE         6+     5399 STRING, NULL TERMINATES STRING
```

```
219C  EB              X218C  XCHG  SAVE PTR TO CHAR BUFR IN DE (HL USED IN SIGN TEST FOR INTEGER VALUE IN 1B25)
219D  CD 25 1B               CALL  X1B25 PERFORM SIGN TEST ON VALUE IN HOLDING AREA - SET or RESET ZERO FLAG FOR USE IN BAC
2190  EB                     XCHG  PUT PTR TO CHAR BUFR BACK IN HL
2191  F2 9E 21               JP    X219E IF VALUE IS POSITIVE, JMP IMMEDIATELY INTO THE CONVERSION ROUTINE
2194  36 2D                  MVI   M,A'-' PUT A '-' IN CHAR BUFR INSTEAD OF LEADING SPACE
2196  C5                     PUSH  B } SAVE BC HL ON STACK (USED BY 1B9C, DE DON'T CARE)
2197  E5                     PUSH  H }
2198  CD 0C 1B               CALL  X1B0C NEGATE VALUE IN HOLDING AREA - EITHER WAY, RESULT IN HOLDING AREA IS A POSITIVE #
219B  E1                     POP   H } RESTORE BCHL FROM THE STACK
219C  C1                     POP   B }
219D  B4                     ORA   H CLEAR CARRY BIT FOR RAL AT 21A5
219E  23              X219E  INX   H   HL POINTS TO 0422 AFTER INX H - 1ST LOC FOR CONVERTED #
219F  36 30                  MVI   M,A'0' STORE ASCII 0 AT 0422 - ALWAYS START WITH A LEADING ZERO
21A1  3A CD 03               LDA   X03CD LOAD ACC WITH BIT-FLAG BYTE
21A4  57                     MOV   D,A   SAVE BIT-FLAG BYTE IN D (IN CASE BRANCH TO 226A)
21A5  17                     RAL   TEST A7 BIT OF BYTE STORED AT 03CD
21A6  3A A6 03               LDA   X03A6 LOAD ACC WITH TYPE-CODE OF VALUE IN HOLDING AREA
21A9  DA 6A 22  PRINT USING  JC    X226A JMP IF PRINT USING FCN IS ACTIVE - FORMATTED OUTPUT
21AC  CA 62 22  UNFORMATTED 0 JZ   X2262 JMP IF # TO CONVERT IS ZERO - UNFORMATTED OUTPUT - PUT SINGLE '0' IN BUFR
21AF  FE 05                  CPI   H'05' TEST TYPE-CODE OF VALUE IN HOLDING AREA
21B1  EA 08 22  SEPARATE RETURN AT 2269
                             JPE   X2208 JMP IF VALUE IS SNG- OR DBL-PREC FLOATING-PT VALUE - CONVERT VALUE TO CHAR STRING
21B4  01 00 00  UNFORMATTED FLTING PT CONVERSION
                             LXI   B,X0000 SET B&C SO NO COMMA'S or DECIMAL PT IN CONVERTED # STRING
21B7  CD FA 24               CALL  X24FA CONVERT # TO ASCII CHAR STRING STARTING AT 0422
21BA  21 21 04        X218A  LXI   H,X0421 LOAD HL WITH START ADDR OF CHAR STRING
21BD  46                     MOV   B,M   FETCH A CHAR FROM STRING (SIGN CHAR)
21BE  0E 20                  MVI   C,A' ' PUT ASCII SPACE IN C - CONVENIENT PLACE, AVOIDS OTHER MVI'S
21C0  3A CD 03               LDA   X03CD LOAD ACC WITH BIT-FLAG BYTE
21C3  5F                     MOV   E,A   SAVE BIT-FLAG BYTE IN E (USED AGAIN AT 21F2)
21C4  E6 20                  ANI   A' ' TEST THE 20 BIT - IF SET, LEADING '*' FILL (see p. 88)
21C6  CA D1 21               JZ    X21D1 JMP IF NOT DOING LEADING '*' FILL
21C9  78                     MOV   A,B   PUT SIGN CHAR IN ACC
21CA  B9                     CMP   C     TEST SIGN CHAR AGAINST SPACE CHAR
21CB  0E 2A                  MVI   C,A'*' PUT '*' CHAR IN C
21CD  C2 D1 21               JNZ   X21D1 IF TEST AT 21CA FAILED TO MATCH, SIGN CHAR IS '+' or '-', DON'T PUT '*' IN THAT POSITION - SIGN CHAR SAVED IN B
21D0  41                     MOV   B,C SIGN CHAR WAS A SPACE - SO FILL IT WITH AN '*' - DESTROY SIGN SPACE FOR 2200
21D1  71              X21D1  MOV   M,C PUT '␣' or '*' in memory - '*' IF DOING '*' FILL, '␣' OTHERWISE
21D2  D7              END OF # RST   2   SCAN TO NXT NON-SPACE CHAR IN CONVERTED # STRING - SET FLAGS, NON-SPACE CHAR IN ACC
21D3  CA EF 21        STRING JZ    X21EF JMP IF END OF CHAR STRING HAS BEEN FOUND (CHAR = NULL)
21D6  FE 45                  CPI   A'E' }
21D8  CA EF 21        'E'    JZ    X21EF } JMP IF CHAR IS 'E' } NO MORE LEADING '0' SUPPRESSION
21DB  FE 44                  CPI   A'D' }                    } or
21DD  CA EF 21        'D'    JZ    X21EF } JMP IF CHAR IS 'D' } LEADING '*' FILL
21E0  FE 30                  CPI   A'0' } IF CHAR IS A '0' (LEADING 0) REPLACE WITH A SPACE OR '*'
21E2  CA D1 21        '0'    JZ    X21D1 }      (LEADING '0' SUPPRESSION)                              - WHICHEVER IS REQUIRED
21E5  FE 2C                  CPI   A',' } IF CHAR IS A ',' REPLACE WITH A SPACE OR '*'
21E7  CA D1 21        ','    JZ    X21D1 }
21EA  FE 2E                  CPI   A'.' } JMP IF CHAR IS NOT A '.'
21EC  C2 F2 21               JNZ   X21F2 }
21EF  2B              X21EF  DCX   H   BACK-UP PTR TO CONVERTED # IN BUFR
21F0  36 30                  MVI   M,A'0' PUT A SINGLE '0' CHAR IN BUFR (BEFORE END OF BUFR, 'E', 'D', or '.')
21F2  7B              X21F2  MOV   A,E } PUT BIT-FLAG BYTE IN ACC & TEST THE 10 BIT
21F3  E6 10                  ANI   H'10' } (=1 IF $ IN FORMAT)
21F5  CA FB 21               JZ    X21FB JMP IF $ FORMAT NOT BEING USED
21F8  2B                     DCX   H   BACK-UP PTR TO CHAR BEFORE LEADING SIGNIFICANT CHAR IN BUFR
21F9  36 24                  MVI   M,A'$' PUT A '$' CHAR IN BUFR BEFORE CONVERTED-# STRING
21FB  7B              X21FB  MOV   A,E } TEST THE 04 BIT OF THE BIT-FLAG BYTE
21FC  E6 04                  ANI   H'04' }
21FE  C0        TRAILING SIGN CHAR
                             RNZ
21FF  2B                     DCX   H   BACK UP PTR TO CHAR STRING
2200  70        LEADING SIGN CHAR
                             MOV   M,B PUT SIGN CHAR IN FRONT OF STRING
                             RET
```

0421 '␣' or sign char
0422 ASCII DIGIT HI ORDER
0423

INTEGER # CONVERT - UNFORMATTED
SUPPRESS LEADING ZERO'S
INSERT SIGN CHAR
LEADING '*' FILL '$' INSERT
LEADING '**' FILL
END OF STRING '0' 'E' '.'

```
2201 C9                    RET                                                  I    5460
2202 32 CD 03      X2202   STA  X03CD STA BIT-FLAG BYTE AT FLAG LOC Ø3CD        2M   5461
2205 21 21 04              LXI  H,X0421 } PUT A SPACE AT LOC Ø421 (2 LOC ABOVE OUTPUT !!   5462
2208 3E 20                 MVI  M,A' ' }                                        6    5463 INITIAL CHAR FOR A SUPPRESSED '+' SIGN
220A C9                    RET                  BUFFER AREA)                    I    5464
                                                                                     5465 FLOATING-PT # CONVERT
220B E5            X2208   PUSH H    SAVE PTR TO CHAR BUFR ON STACK (PT TO LOC Ø422)                5466
220C DE 00 CARRY SET FOR   SBI  H'00' SET FLAGS - (CLEARS CARRY BIT) IN ACC DBL-PREC 8  SNG-PREC 3 ^    5466
220E 17    SNG-PREC        RAL  MULTIPLY BY 2  IN ACC (SNG-PREC 6) (DBL-PREC 16)                     5467 UNFORMATTED OUTPUT
220F 57                    MOV  D,A  STORE RESULT IN D                          H    5468
2210 14                    INR  D  ADD 1 TO RESULT (SNG-PREC 7) (DBL-PREC 17) [ONE MORE THAN # OF DIGITS TO PRINT
2211 CD 08 23              CALL X2308 ADJUST # IN HOLDING AREA SUCH THAT (SNG-PREC Ø^-5 ≤ # < Ø^6) or (DBL-PREC Ø^-5 ≤ # < Ø^16) MX#  5470 ACC HAS EXPONENT ADJUSTMENT  X10'S -> - ACC
2214 01 00 03               LXI  B,X0300 LOAD DECIMAL-PT POSITION CTR WITH Ø3 (B), SET COMMA POSITION CTR TO Ø  5471 (NO COMMA'S)  ÷10'S -> + ACC
2217 82                    ADD  D  } TEST DECIMAL EXPONENT-ADJUST VALUE (OBTAINED FROM 2308)                  5472
2218 FA 24 22 SNG #<10^-5  JM   X2224 } JMP IF # ORIGINAL VALUE (SNG-PREC # < 10^-5) or (DBL-PREC # < 10^-3)  $"   5473
221B 14    DBL #<10^-3     INR  D  } TEST DECIMAL EXPONENT-ADJUST VALUE AGAIN                                 5474
221C BA    ACC IS NEG #    CMP  D  } JMP IF # ORIGINAL VALUE (SNG-PREC # ≥ 10^6) or (DBL-PREC # ≥ 10^16)  <    5475
221D D2 24 22 SNG #≥10^6   JNC  X2224                                           R$"  5476
2220 3C    DBL #≥10^16     INR  A ADD 1 TO COMPENSATE FOR DCR B AT START OF LOOP AT 2451 <    5477
2221 47    ACC IS POS #    MOV  B,A PUT NEW VALUE FOR DECIMAL-PT POSITION CTR IN B      G    5478
2222 3E 02                 MVI  A,H'02' LOAD ACC WITH Ø2 SO ZERO FLAG WILL BE SET BY NXT INSTRUCTION TO >FORCE 5479 NON-EXPONENT OUTPUT FORMAT
2224 D6 02         X2224   SUI  H'02' SET OR CLEAR ZERO FLAG - IF RESET, USE EXPONENTIAL FORMAT  V    5480
2226 E1                    POP  H FETCH PTR TO CHAR BUFR FROM STACK                              5481
2227 F5                    PUSH PSW SAVE FLAG STATUS ON STACK                                    5482
2228 CD 51 24 ONE '0' IN BUFR FROM 219F  CALL X2451 INSERT DP IN CHAR BUFR IF NECESSARY AT LOC Ø422 (KILLS LEADING MQ$  5483 ZERO PUT IN AT 219F)
222B 36 30                 MVI  M,A'0'                                          60   5484
222D CC 5B 19              CZ   X195B                                           L[   5485
2230 CD 55 24              CALL X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT  M%  5486 - IF INTEGER, DP LAST CHAR IN BUFR
2233 2B            X2233   DCX  H BACK-UP PTR TO CHAR BUFR                       +    5487
2234 7E                    MOV  A,M FETCH CHAR FROM CONVERTED #                        5488
2235 FE 30                 CPI  A'0' TEST CHAR                                  0    5489
2237 CA 33 22              JZ   X2233 JMP IF CHAR IS 'Ø' - LOOP UNTIL ALL TRAILING 'Ø' CHARS TO RIGHT OF DP J3"DP 5490 HAVE BEEN REMOVED
223A FE 2E                 CPI  A'.' TEST CHAR                                  .    5491
223C C4 5B 18              CNZ  X185B IF CHAR IS NOT '.', LEAVE CHAR IN BUFR AS PART OF RESULT BY DOING D[ANY 5492 INX H
223F F1                    POP  PSW FETCH FLAG STATUS FROM STACK                       5493
2240 CA 63 22 NO EXPONENT TO CONVERT  JZ  X2263 JMP IF (SNG PREC Ø^-5 ≤ # < Ø^6) or (DBL PREC Ø^-3 ≤ # < Ø^16) (EXPONENT FORMAT J NOT 5494 BEING USED)
2243 F5            X2243   PUSH PSW SAVE FLAG STATUS ON STACK & EXPONENT VALUE                  5495  ↓ CONVERT + PRINT EXPONENT OF #
2244 F7                    RST  6  TEST TYPE OF VALUE IN HOLDING (# TO BE CONVERTED) RNC - DBL-PREC 5496 RC - SNG-PREC
2245 3E 22                 MVI  A,A'"' LOAD ACC WITH 22 (½ OF 'D')              >"   5497
2247 8F                    ADC  A  COMPUTE A CHAR, IF # IS DBL-PREC, CHAR IS 'D', IF # IS SNG-PREC,  5498 CHAR IS 'E'
2248 77                    MOV  M,A  PUT CHAR FOR EXPONENT IN CHAR BUFR                5499
2249 23                    INX  H  ADVANCE PTR TO CHAR BUFR                     #    5500
224A F1                    POP  PSW FETCH FLAG STATUS FROM STACK & EXPONENT VALUE      5501
224B 36 2B                 MVI  M,A'+' PUT A '+' IN CHAR BUFR FOR EXPONENT      6+   5502
224D F2 54 22              JP   X2254 IF EXPONENT IS POSITIVE, JMP TO 2254 TO CONVERT EXPONENT  I"  5503 MAGNITUDE
2250 36 2D                 MVI  M,A'-' PUT A '-' IN CHAR BUFR FOR EXPONENT      6-   5504
2252 2F                    CMA     } FORM 2'S COMPLEMENT OF EXPONENT MAGNITUDE  /    5505
2253 3C                    INR  A  } (POSITIVE # IN ACC)                        <    5506
2254 06 2F         X2254   MVI  B,A'/' SET CHAR INITIALLY TO '/' (1 LESS THAN 'Ø')  /    5507
2256 04            X2256   INR  B  ADVANCE CHAR COUNT BY 1                             5508
2257 D6 0A                 SUI  H'0A' SUBTRACT 10 FROM EXPONENT MAGNITUDE       V    5509
2259 D2 56 22              JNC  X2256 LOOP UNTIL EXPONENT MAGNITUDE IS LESS THAN 10 RV"  5510
225C C6 3A                 ADI  A':' CONVERT REMAINDER IN ACC TO A CHAR ('Ø'-'9') - ADD 1 MORE THAN '9' F:TO  5511 # IN ACC
225E 23                    INX  H  ADVANCE PTR TO CHAR BUFR                     #    5512
225F 70                    MOV  M,B PUT 1ST CHAR OF EXPONENT MAGNITUDE IN CHAR BUFR    5513
2260 23                    INX  H  ADVANCE PTR TO CHAR BUFR                     #    5514
2261 77                    MOV  M,A PUT 2ND CHAR OF EXPONENT MAGNITUDE IN CHAR BUFR    5515
2262 23            X2262   INX  H  ADVANCE PTR TO CHAR BUFR                     #    5516  ↓ TERMINATE CHAR BUFFER
2263 36 00         X2263   MVI  M,H'00' PUT A NULL IN CHAR BUFR TO TERMINATE CONVERTED # STRING  6  5517
2265 EB                    XCHG PUT ADDR OF NULL OF CONVERTED # STRING IN DE (USED BY 2303)    5518
2266 21 21 04              LXI  H,X0421 LOAD HL WITH ADDR OF 1ST CHAR OF CONVERTED #  !!   5519
                           RET
```

```
2259  C9                        [illegible]
225A  23          X226A   INX   H ADVANCE PTR TO LOC 0423 (LOC AFTER 1ST LEADING '0')
225B  C5                  PUSH  B PUT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP (B) & # OF NUMERIC FIELD POSITIONS TO
225C  FE 04               CPI   H'04' TEST VALUE TYPE CODE OF # TO CONVERT (PUT IN ACC AT 21A6)
225F  7A                  MOV   A,D  PUT BIT FLAG BYTE IN ACC
226F  D2 E0 22            JNC   X22E0 JMP IF VALUE TO CONVERT IS FLOATING-PT (SNG-OR DBL-PREC)
2272  1F                  RAR   TEST 01 BIT OF BIT-FLAG BYTE IN ACC. (see p. 88)
2273  DA 78 23            JC    X2378 JMP IF INTEGER VALUE TO BE PRINTED IN EXPONENTIAL FORMAT
2276  01 03 06            LXI   B,X0603 LOAD B WITH A DIGIT COUNT FOR DP POSITION (5 DIGITS +1=COUNT), LOAD C WITH
2279  CD 49 24            CALL  X2449 TEST BIT-FLAG FOR COMMA FORMAT IN CONVERTED # - CLEAR C IF NO COMMA'S
227C  D1                  POP   D PUT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP IN D
227D  7A                  MOV   A,D  } COMPUTE # OF NUMERIC FIELD POSITIONS THAT CANNOT BE USED BY
227E  D6 05               SUI   H'05'} CONVERTED #
2280  F4 25 24            CP    X2425 IF ANY SPARE FIELD POSITIONS, FILL THEM WITH LEADING '0' CHAR'S
2283  CD FA 24            CALL  X24FA CONVERT VALUE TO DIGIT STRING USING 2 BYTE SUBTRACT ROUTINE
2286  78 FROM 232A X2286  MOV   A,E  PUT # OF NUMERIC FIELD POSITIONS TO RIGHT OF DP (INCLUDES DP POSITION) IN ACC.
2287  B7                  ORA   A    TEST FIELD COUNT
2288  CC C7 1A            CZ    X1AC7 IF COUNT IS 0, REMOVE '.' CHAR IN BUFR BY DCX H (PUT IN BUFR BY
2280  3D                  DCR   A DECREMENT RIGHT-FIELD COUNT (COUNTS 1 POSITION FOR DP)
228C  F4 25 24            CP    X2425 FILL BUFR WITH TRAILING '0' CHAR'S - # OF '0'S IN ACC FROM RIGHT
228F  E5          X228F   PUSH  H  SAVE PTR TO NXT BYTE AVAILABLE IN CHAR BUFR ON STACK - END OF STRING PTR
2290  CD BA 21            CALL  X21BA PERFORM LEADING '0' SUPPRESSION, INSERT '#', '$', + SIGN CHAR IN FRONT OF
2293  E1                  POP   H  FETCH PTR TO BUFR INTO HL - POINTS TO NXT BYTE AFTER #-STRING
2294  CA 99 22            JZ    X2299 JMP IF SIGN CHAR BELONGS IN FRONT OF # (see 21FF - 2200) -
2297  70                  MOV   M,B STORE SIGN CHAR AFTER CONVERTED #
2298  23                  INX   H  ADVANCE PTR TO NXT BYTE AVAILABLE IN CHAR BUFR
2299  36 00     X2299     MVI   M,H'00' STORE A NULL IN BUFR TO TERMINATE THE CHAR STRING
229B  21 20 04            LXI   H,X0420 LOAD HL WITH ADDR OF BYTE BEFORE 1ST CHAR OF CONVERTED #
229E  23          X229E   INX   H  ADVANCE PTR TO NXT CHAR IN BUFR
229F  3A 05 03    X229F   LDA   X0305 LOAD ACC WITH LO-ORDER 8 BITS OF ADDR WHERE '.' CHAR IS STORED
22A2  95                  SUB   L SUBTRACT LO-ORDER BYTE OF ADDRESS OF 1ST CHAR IN BUFR
22A3  92                  SUB   D SUBTRACT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP (see 227C)
22A4  C8          DONE    RZ    CONVERSION IS DONE IF # OF DIGITS TO LEFT OF DP MATCHES FORMAT
22A5  7E                  MOV   A,M FETCH CHAR FROM CONVERTED # STRING INTO ACC
22A6  FE 20               CPI   A' ' } TEST CHAR
22A8  CA 9E 22            JZ    X229E} JMP IF CHAR IS '␣'
22AB  FE 2A               CPI   A'*' } TEST CHAR
22AD  CA 9E 22            JZ    X229E} JMP IF CHAR IS '*'
2280  2B                  DCX   H  BACK-UP PTR TO LAST '␣' or '*' (CHAR BEFORE 1ST DIGIT, '$', or '+' or '-')
2281  E5                  PUSH  H  SAVE PTR ON STACK
2282  F5          X2282   PUSH  PSW SAVE CHAR ON STACK - ZERO FLAG SET (1ST TIME THRU ZERO FLAG RESET)
2283  01 82 22            LXI   B,X2282} PUT ADDR 2282 ON STACK FOR IMPLIED JMP
2286  C5                  PUSH  B
2287  D7                  RST   2 SCAN TO NXT NON-SPACE CHAR IN STRING - SET FLAGS - CHAR IN ACC
2288  FE 2D               CPI   A'-' } TEST CHAR & JMP IF CHAR IS '-'
228A  C8                  RZ
2289  FE 2B               CPI   A'+' } TEST CHAR & JMP IF CHAR IS '+'
2290  C8                  RZ
228E  FE 24               CPI   A'$' } TEST CHAR & JMP IF CHAR IS '$'
22C0  C8                  RZ
22C1  C1                  POP   B  REMOVE THE IMPLIED JMP ADDR FROM THE STACK
22C2  FE 30               CPI   A'0' TEST CHAR
22C4  C2 C8 22            JNZ   X22C8 JMP IF CHAR IS NOT A LEADING '0' - REMOVE CHAR'S FROM STACK & PUT
22C7  23                  INX   H  ADVANCE PTR TO BUFR } i.e. ␣␣+$00 1,234
22C8  D7                  RST   2 SCAN TO NXT CHAR     }
22C9  D2 08 22            JNC   X2208 JMP IF CHAR IS NOT A DIGIT
22CC  2B                  DCX   H  BACK-UP PTR (CANCELS INX H AT 22C7)
22CD  01 2B 77    DCX H MOV M,A LXI B,X772B} PUT SIGN CHAR +/or $ IN BUFR FROM STACK
22D0  F1                  POP   PSW        }
22D1  CA CE 22            JZ    X22CE      } (PUT ON TOP OF '0' FOUND AT 22C2 - ELIMINATES ONE LEADING '0')
```

```
5520  I
5521  0
5522  E RIGHT OF DP (INCLUDING DP) (C) ON STACK (see 1671-1673)
5523
5524
5525  R ''
5526  INTEGER VALUE CONVERT-FORMATTED OUTPUT
5527  Z 0
5528  DIGIT COUNT FOR COMMA POSITION
5529  IN FORMAT
5530  Q
5531  BY
5532  V
5533  %$
5534  M $
5535
5536  7
5537  LG CALL 2451 AT 2528, 2326, or 2432)
5538  =
5539  FIELD COUNT
5540
5541  M !#- STRING (IF FORMATTED THAT WAY)
5542
5543  J '' SIGN CHAR PUT IN FRONT OF # AT 21FF-2200
5544
5545  #
5546  6
5547  !
5548  #
5549  !U
5550
5551
5552  H>
5553
5554  TEST FOR ENOUGH CHAR'S TO FILL
5555  J ''
5556  *
5557  J ''
5558  +  -
5559
5560
5561  2''
5562  E
5563  H
5564  -
5565  H
5566  +
5567  H
5568  $
5569  H
5570
5571  0
5572  0X'' A
5573  #
5574  H
5575  RX''
5576
5577  +
5578
5579  JN'' Ø')
```

IN D.
XXXXXXX.XX
NEEDED FOR INTEGER VALUE

ADJUST HL TO POINT TO 1ST CHAR OF CONVERTED # STRING ACCORDING TO FORMAT REQUIREMENTS
D = # OF DIGITS LEFT OF DP
␣␣XXX.XX
(HL) (0306 0305)

THIS IS WHERE '0' IS PUT FOR OVERFLOW INDICATION

PUT SIGN +/or $ ON STACK PRIOR TO REMOVING A LEADING 0

'0' IN BUFR

PUT ON STACK (HL) INX H CHAR FOUND BY RST 2 AT 22D2 AT LEAST 1 DIGIT IN FRONT OF DP

```
2204  C1                POP   B REMOVE PTR ADDR FROM STACK (NOT NEEDED HERE)
2205  C3 9F 22          JMP   X229F LOOP UNTIL ENOUGH LEADING Ø'S HAVE BEEN REMOVED
2208  F1          X2208 POP   PSH  REMOVE A CHAR FROM STACK
2209  CA C8 22          JZ    X2208 LOOP UNTIL CHAR REMOVED WITH ZERO FLAG RESET (see 22AD-22B2)
220C  E1                POP   H  FETCH PTR TO CHAR POSITION IN BUFR BEFORE 1ST SIGNIFICANT CHAR (DIGIT '1'-'9')
220D  36 25             MVI   M,A'%' PUT '%' CHAR IN BUFR TO INDICATE OVERFLOW
220F  C9                RET   CONVERSION DONE
```

```
                                                                    A     5580
                                                                    C "   5581
                                                                          5582
                                                                    JX"   5583  ↘ FORMAT FIELD OVERFLOW
                                                                          5584
                                                                    6%    5585
                                                                    I     5586
                                                                          5587
```

```
                                                                          5588  FLOATING-PT VALUE CONVERT
                                                                          5589  FORMATTED OUTPUT
                                                                          5590
                                                                    I%    5591
                                                                    M\    5592
                                                                          5593
                                                                    #     5594
                                                                          5595  ↘ MAGNITUDE OF # TO CONVERT TOO LARGE FOR FIELD (NON-EXPONENTIAL FORMAT)
                                                                          5596  DP (INCLUDING DP) IN C (see 226B)
                                                                    M !   5597
                                                                    +     5598
                                                                    6%    5599
                                                                    I     5600
```

```
22E0  E5          X22E0 PUSH H SAVE PTR TO LOC Ø423 ON STACK (LOC AFTER 1ST LEADING 'Ø')
22E1  1F                RAR   TEST Ø1 BIT OF BIT-FLAG BYTE BY SHIFTING RIGHT 1 PLACE - CARRY BIT NOT SET UPON ENTRY
22E2  DA 82 23          JC    X2382 JMP IF DOING EXPONENTIAL FORMAT (Ø1 BIT SET IN BIT-FLAG BYTE - see 2B88)
22E5  CA FC 22          JZ    X22FC JMP IF VALUE TO CONVERT IS SNG-PREC VALUE (RAR NOT AFFECT Ø-FLAG - see 2B6C)
22E8  11 49 25          LXI   D,X2549 LOAD DE WITH ADDR OF DBL-PREC VALUE = 1Ø16
22EB  CD CC 19          CALL  X19CC PERFORM DBL-PREC MAGNITUDE COMPARISON
22EE  16 10             MVI   D,H'10' PUT A DIGIT COUNT OF 16 IN D FOR DBL-PREC CONVERT
22F0  FA 0A 23          JM    X230A JMP IF # TO CONVERT IS < 1Ø16 - # HAS LESS THAN 17 DIGITS TO PRINT
22F3  E1          X22F3 POP   H  PUT PTR TO LOC Ø423 IN HL (see 22EØ)
22F4  C1                POP   B  PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN B, PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN C (see 226B)
22F5  CD 81 21          CALL  X2181
22F8  2B                DCX   H  BACK-UP PTR TO CHAR BUFR (CHAR POSITION BEFORE SIGN CHAR OR '$')
22F9  36 25             MVI   M,A'%' PUT '%' CHAR IN BUFR TO INDICATE OVERFLOW
22FB  C9                RET
```

```
                                                                    M !   5597
                                                                    +     5598
                                                                    6%    5599
                                                                    I     5600
```

EXPONENTIAL FORMAT

```
22FC  01 0E 86    X22FC LXI   B,X860E } LOAD BCDE WITH SNG-PREC FLOATING-PT VALUE = 1Ø16
22FF  11 CA 13          LXI   D,X13CA }
2302  CD A1 18          CALL  X18A1 PERFORM SNG-PREC MAGNITUDE COMPARISON
2305  F2 F3 22          JP    X22F3 JMP IF #-TO-CONVERT ≥ 1Ø16 - FIELD OVERFLOW - LARGEST NON-EXPONENTIAL FIELD IS 16 DIGITS
2308  16 06             MVI   D,H'06' PUT A DIGIT COUNT OF 6 IN D FOR SNG-PREC CONVERT
230A  EF          X230A RST   5 TEST SIGN OF #-TO-CONVERT IN HOLDING AREA
230B  C4 C8 23          CNZ   X23C8 IF # ≠ Ø, ADJUST # IN HOLDING AREA SUCH THAT (SNG-PREC 1Ø5 ≤ # < 1Ø6) or (DBL-PREC 1Ø15 ≤ # < 1Ø16)
230E  E1                POP   H  PUT PTR TO LOC Ø423 IN HL (see 22EØ)
230F  C1                POP   B  PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN B, PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN C (see 226B)
2310  FA 20 23          JM    X2320 JMP IF # TO CONVERT WAS MULTIPLIED BY 1Ø (N TIMES) BY 23D8
2313  C5                PUSH  B  PUT # OF NUMERIC POSITIONS TO LEFT OF DP (B) & # OF NUMERIC POSITIONS TO RIGHT OF DP (C) ON STACK
2314  5F                MOV   E,A PUT EXPONENT ADJUSTMENT IN E (FROM CALL 23D8)
2315  79                MOV   A,B } COMPUTE # OF LEADING ZERØ'S NEEDED TO FILL ALL NUMERIC POSITIONS TO LEFT OF DP
2316  92                SUB   D   } # IN ACC = (LEFT NUMERIC COUNT) - (DIGIT COUNT) - (EXPONENT ADJUSTMENT)
2317  93                SUB   E   }                                                    (+#)
2318  F4 25 24          CP    X2425 IF ACC IS ≥ Ø, FILL BUFR WITH N LEADING Ø'S
231B  CC 3C 24          CALL  X243C COMPUTE VALUES FOR DECIMAL-PT & COMMA POSITION CTR'S
231E  CD 65 24          CALL  X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT
2321  B3                ORA   E  PUT EXPONENT ADJUSTMENT IN ACC (see 2314) - set FLAGS
2322  C4 35 24          CNZ   X2435 PUT TRAILING 'Ø'S IN CHAR BUFR (AS MANY AS NECESSARY)
2325  B3                ORA   E  PUT EXPONENT ADJUSTMENT IN ACC - set FLAGS
2326  C4 51 24          CNZ   X2451 IF CHAR ('Ø') PUT IN BUFR AT 2322, TEST IF COMMA OR DP GOES IN BUFR NEXT
2329  D1                POP   D  PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN D, PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN E
232A  C3 96 22          JMP   X2286 CONVERSION ALMOST DONE - CLEAN UP STRING, LEADING Ø' SUPPRESSION, INSERT SIGN CHAR
```

```
                                                                    M<$   5616  RESULT: (B) - (D) - (E) = 3
                                                                    M-$   5617
                                                                    3     5618  ØØØDDDDDDØ
                                                                    D5$   5619  ORIGINAL #
                                                                    3     5620  in SNG-PREC
                                                                    DQ$   5621
                                                                    Q     5622
                                                                    C "   5623
                                                                          5624
                                                                          5625
```

```
2320  5F          X2320 MOV   E,A PUT EXPONENT ADJUSTMENT IN E (FROM CALL 23D8)
232C  79                MOV   A,C PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN ACC
232D  B7                ORA   A  TEST RIGHT-FIELD CTR
232E  C4 05 20          CNZ   X2005 IF COUNT IS NOT Ø, DCR ACC BY 1 (ACCOUNTS FOR DP - TEMPORARY COMPUTATION see 2373)
2331  83          (-#)  ADD   E  ADD EXPONENT ADJUSTMENT (# OF DIGITS OF ORIGINAL # THAT SHOULD BE ON RIGHT OF DP)
2332  FA 38 23          JM    X2338 IF SUM IS MINUS, # OF DIGITS IN ORIGINAL # TO RIGHT OF DP EXCEEDS FORMAT POSITIONS AVAILABLE
2335  AF                XRA   A  CLEAR ACC - NO LOSS OF DIGITS OR ROUND-OFF INVOLVED - ENOUGH ROOM TO RIGHT OF DP FOR ALL DIGITS
2336  C5          X2338 PUSH  B  SAVE BC ON STACK
2337  F5                PUSH  PSW SAVE LOST-DIGIT COUNT ON STACK (NEGATIVE #)
2338  FC 07 20    X233A CM    X2007 DIVIDE # BY 1Ø, ADD 1 TO ACC (PUSH 1 DIGIT TO RIGHT OF DP TO BE LOST IN CALCULATION)
233B  FA 3A 23          JM    X233A LOOP UNTIL ENOUGH LOST DIGITS HAVE BEEN PUSHED TO RIGHT OF DP
233E  C1                POP   B  PUT LOST-DIGIT COUNT IN B
233F  79                MOV   A,E PUT EXPONENT ADJUSTMENT IN ACC (NEGATIVE #)
2340  90                SUB   B  SUBTRACT LOST-DIGIT COUNT (NEGATIVE) or Ø
2341  C1                POP   B  RESTORE BC FROM STACK
```

```
                                                                    A     5580
                                                                    C "   5581
                                                                          5582
                                                                    JX"   5583  ↘ FORMAT FIELD OVERFLOW
                                                                          5584
                                                                    6%    5585
                                                                    I     5586
                                                                          5587
                                                                          5588  FLOATING-PT VALUE CONVERT
                                                                          5589  FORMATTED OUTPUT
                                                                          5590
                                                                    I%    5591
                                                                    M\    5592
                                                                          5593
                                                                    #     5594
                                                                          5595  ↘ MAGNITUDE OF # TO CONVERT TOO LARGE FOR FIELD (NON-EXPONENTIAL FORMAT)
                                                                          5596  DP (INCLUDING DP) IN C (see 226B)
                                                                    M !   5597
                                                                    +     5598
                                                                    6%    5599
                                                                    I     5600
                                                                    6     5601
                                                                    J     5602
                                                                    M!    5603
                                                                          5604  FIELD IS 16 DIGITS
                                                                          5605
                                                                          5606  ACC HAS EXPONENT ADJUSTMENT  X1Ø'S => -ACC
                                                                    DX0   5607                             ÷1Ø'S => +ACC
                                                                          5608
                                                                    A     5609  (INCLUDING DP) IN C (see 226B)
                                                                    -#    5610  ↘ #'S ≥ 1Ø5 or 1Ø15 - NO FRACTIONS
                                                                    E     5611  (INCLUDING DP) (C) ON STACK
                                                                    -     5612  in SNG-PREC
                                                                          5613  ORIGINAL # DDDDDD.DD
                                                                          5614  AFTER 23D8 DDDDDD.DDD ADJUSTMENT = 1 (E)
                                                                          5615  # IN D (= Ø6 FOR SNG-PREC)
                                                                          5616  ROUND-OFF OF 10 DIGITS
                                                                          5617
                                                                          5618  ØØØDDDDDDØ
                                                                          5619  ORIGINAL #
                                                                          5620  in SNG-PREC
                                                                          5621
                                                                          5622
                                                                          5623
                                                                          5624
                                                                          5625
                                                                          5626  ↘ #'S < 1Ø5 or 1Ø15
                                                                          5627  in SNG-PREC
                                                                    CU    5628  original #  DDDD.DDDD  ADJUSTMENT = +2
                                                                          5629  DDDDDD.DD (E)
                                                                          5630  AFTER ADJUSTMENT # IN D (Ø6 FOR SNG-PREC)
                                                                          5631  ROUND OFF OF 10 DIGITS
                                                                    E     5632
                                                                          5633  ORIGINAL # DD.DDDD ADJUST DDDDDD.
                                                                          5634  FORMAT XXXX.XXX
                                                                    W     5635  DIVIDE # ONCE  DDDDD.D
                                                                    A     5636  ↖ THIS DIGIT LOST IN CONVERSION ROUNDED-OFF
                                                                          5637
                                                                          5638
                                                                    A     5639
```

```
2344  5F                    MOV  E,A PUT NEW VALUE FOR EXPONENT ADJUSTMENT IN E (LOSING DIGITS IS A READJUSTMENT TO EXPONENT)
2345  82                    ADD  D  =Ø6 FOR SNG-PREC =1ØH FOR DBL-PREC
2346  78                    MOV  A,B PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN ACC
2347  FA 56 23              JM   X2356 JMP IF ENTIRE # IS SMALL ENOUGH TO FIT ENTIRELY TO RIGHT OF DP - ALL 6 V#  DIGITS OF #
234A  92                    SUB  D } COMPUTE # OF LEADING ZERO'S NEEDED TO FILL ALL NUMERIC POSITIONS TO LEFT OF
234B  93                    SUB  E }   DP     # IN ACC
234C  F4 25 24              CP   X2425 IF ACC ≥ Ø, FILL BUFR WITH N LEADING Ø'S                          %$
234F  C5                    PUSH B PUT # OF NUMERIC POSITIONS TO LEFT OF DP (B) & # OF POSITIONS TO RIGHT OF DP (INCLUDING DP) ON STACK
2350  CD 3C 24              CALL X243C COMPUTE VALUES FOR DECIMAL-PT & COMMA POSITION CTR'S              M<$
2353  C3 67 23              JMP  X2367 SKIP AROUND NXT INSTRUCTIONS                                     C #
2356  CD 25 24  X2356       CALL X2425 FILL ALL PLACES ON LEFT OF DP WITH N LEADING Ø'S                  M%$
2359  79                    MOV  A,C  PUT RIGHT NUMERIC POSITION COUNT IN ACC (SINCE C GETS DESTROYED BY NXT INSTR)
235A  CD 55 24              CALL X2455 PUT '.' IN CHAR BUFR                                             MU$
235D  4F                    MOV  C,A PUT RIGHT NUMERIC POSITION COUNT IN C AGAIN                        0
235E  AF                    XRA  A } COMPUTE # OF Ø'S NEEDED TO RIGHT OF DP BEFORE # STRING             /
235F  92                    SUB  D }
2360  93                    SUB  E }
2361  CD 25 24              CALL X2425 FILL BUFR WITH N MORE Ø'S                                        M%$
2364  C5                    PUSH B PUT # OF LEFT NUMERIC POSITIONS (B) & # OF RIGHT NUMERIC POSITIONS (C) ON STACK  E
2365  47                    MOV  B,A } CLEAR B & C - NO COMMA'S OR DP TO BE PUT IN BUFR IN THIS CASE     G
2366  4F                    MOV  C,A } (ACC = Ø FROM 2361)                                              0
2367  CD 65 24  X2367       CALL X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT - ØØ IN ACC  M/n$  (see 252E)
236A  C1                    POP  B PUT # OF LEFT NUMERIC POSITIONS IN B, PUT # OF RIGHT NUMERIC POSITIONS (INCLUDING DP) IN C  A
236B  B1                    ORA  C PUT RIGHT NUMERIC COUNT IN ACC - SET FLAGS  (POSITIVE #)             1
236C  C2 72 23              JNZ  X2372 JMP IF RIGHT NUMERIC COUNT ≠Ø                                    B #
236F  2A 05 03              LHLD X0305 IF RIGHT NUMERIC COUNT = Ø, LAST CHAR IN BUFR IS '.', THIS IS WHERE NULL IS TO BE  *U
2372  83                    ADD  E ADD EXPONENT ADJUSTMENT (NEGATIVE #)                                 PUT AT END OF BUFR (see 2299)
2373  3D                    DCR  A DECREMENT RIGHT POSITION COUNT (ACCOUNTS FOR DP)                     =
2374  F4 25 24              CP   X2425 IF ACC ≥ Ø, FILL BUFR WITH N TRAILING Ø'S                        %$
2377  50                    MOV  D,B PUT # OF LEFT NUMERIC POSITIONS IN D                               P
2378  C3 9F 22              JMP  X229F CONVERSION ALMOST DONE - CLEAN UP STRING, LEADING ZERO SUPPRESSION, INSERT  C$,-
237B  E5        X237B       PUSH H SAVE PTR TO LOC Ø423 ON STACK (LOC AFTER 1ST LEADING 'Ø')
237C  D5                    PUSH D SAVE BIT-FLAG BYTE ON STACK (PUT IN D AT 21A4)                        U
237D  CD 60 1C              CALL X1C60 CONVERT INTEGER VALUE TO SNG-PREC VALUE                           M
2380  D1                    POP  D PUT BIT-FLAG BYTE IN D                                               Q
2381  AF                    XRA  A CLEAR ACC TO FORCE JMP AT NXT INSTRUCTION                            /
2382  CA 88 23  X2382       JZ   X2388 JMP IF CONVERTING INTEGER OR SNG-PREC # TO EXPONENTIAL FORMAT     J #
2385  1E 10                 MVI  E,H'10' PUT A DIGIT COUNT OF 1ØH (=16) IN E FOR DBL-PREC CONVERT
2387  01 1E 06    MVI E,'Ø6' LXI  B,X061E DUMMY PUT A DIGIT COUNT OF Ø6 IN E FOR SNG-PREC CONVERT
238A  EF                    RST  5 TEST SIGN OF #-TO-CONVERT IN HOLDING AREA
238B  37                    STC  SET CARRY - USED TO INDICATE #-TO-CONVERT IS Ø - CARRY IS CLEARED BY CALL 2370B IF IT IS EXECUTED
238C  C4 0B 23              CNZ  X230B IF # ≠ Ø, ADJUST # IN HOLDING AREA SUCH THAT (SNG-PREC 1Ø^5 ≤ # < 1Ø^6) or DBL-PREC  DX#
238F  F1                    POP  H PUT PTR TO LOC Ø423 IN HL (see 22E3 or 2378)                         ACC HAS EXPONENT ADJUSTMENT
2390  C1                    POP  B PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN B, PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN C (see 276B)
2391  F5                    PUSH PSW SAVE EXPONENT ADJUSTMENT & FLAG STATUS ON STACK
2392  79                    MOV  A,C } PUT RIGHT NUMERIC COUNT IN ACC
2393  B7                    ORA  A   } & TEST IT                                                        7
2394  F5                    PUSH PSW SAVE FLAG STATUS FOR LATER USE
2395  C4 05 20              CNZ  X2005 IF COUNT IS NOT Ø, DCR ACC BY 1 (ACCOUNTS FOR DP IN RIGHT NUMERIC COUNT)  DU
2398  80                    ADD  B  ADD LEFT NUMERIC COUNT TO ACC
2399  4F                    MOV  C,A STORE TOTAL NUMERIC POSITION COUNT IN C                            0
239A  7A                    MOV  A,D } TEST BIT FLAG BYTE FOR TRAILING SIGN CHAR FORMAT
239B  E6 04                 ANI  H'04' }   (Ø4 BIT SET IF TRAILING SIGN CHAR TO BE USED)
239D  FE 01                 CPI  H'01' }
239F  9F                    SBB  A  } RESULT: ACC = -1 IF LEADING SIGN  ACC = Ø IF TRAILING SIGN  (TRAILING
23A0  57                    MOV  D,A SAVE THIS SIGN COUNT IN D. (-1 FOR LEADING SIGN Ø OTHERWISE)       SIGN NOT PART OF NUMERIC POSITION
23A1  81                    ADD  C } COMPUTE # OF NUMERIC POSITIONS (TOTAL)                             COUNTS (see p 88
23A2  4F                    MOV  C,A } STORE THIS NEW VALUE BACK INTO C   XXX.XX                        0
23A3  93                    SUB  E  SUBTRACT DIGIT COUNT FROM TOTAL COUNT (DIGIT COUNT = Ø6 FOR SNG-PREC, = 1Ø FOR DBL-PREC)
23A4  F5                    PUSH PSW SAVE FLAG STATUS FOR LATER USE (IF ACC IS MINUS, NOT ENOUGH ROOM IN FORMAT FOR ALL DIGITS, ACC HAS -# OF DIGITS TO
                                                                    (IF ≥ Ø  EXTRA Ø'S TO BE ADDED (TRAILING ZEROS))
```

ACC ≠ Ø ZERO FLAG SET IF SNG-PREC (see 2260)

INTEGER # IN EXPONENTIAL FORMAT

FLOATING-PT # IN EXPONENTIAL FORMAT

ØØØ, ØØXXXXXX

12

```
23A5  C5                    PUSH B SAVE BC ON STACK (THIS PAIR NOT SAVED ON STACK BY 2ØD7)
23A6  FC D7 20    X23A6     CM   X20D7 IF ACC HAS -#, DIVIDE ADJUSTED # (see 238C) BY 1Ø - ELIMINATES 1 DIGIT
23A9  FA A6 23              JM   X23A6 LOOP UNTIL ACC=Ø, N DIGITS HAVE BEEN REMOVED FROM # BEFORE
23AC  C1                    POP  B RESTORE BC FROM STACK
23AD  F1                    POP  PSW FETCH FLAG STATUS FROM STACK (-# IN ACC IF NOT ALL DIGITS OF # TO BE
23AE  C5                    PUSH B SAVE BC ON STACK AGAIN      (+# >Ø IF EXTRA Ø'S NEEDED)
23AF  F5                    PUSH PSW SAVE FLAG STATUS & LOST DIGIT COUNT (GAINED DIGIT COUNT IF >Ø)
23B0  FA B4 23              JM   X23B4 JMP IF NOT ALL DIGITS IN # TO BE PRINTED AFTER CONVERSION
23B3  AF                    XRA  A CLEAR ACC - NO DIGITS LOST
23B4  2F          X23B4     CMA  } FORM 2'S COMPLEMENT OF # OF DIGITS LOST (+# IN ACC NOW)
23B5  3C                    INR  A } SAME AS # OF TIMES 23A6-23A9 LOOP EXECUTED
23B6  80                    ADD  B ADD LEFT NUMERIC COUNT TO ACC
23B7  3C                    INR  A ADD 1 TO ACC (COMPENSATES FOR DCR B AT START OF LOOP AT 2451)
23B8  82                    ADD  D ADD SIGN CHAR COUNT (-1 FOR LEADING SIGN, Ø OTHERWISE)
23B9  47                    MOV  B,A PUT VALUE FOR DECIMAL-PT POSITION CTR IN B (USED AT 2451)
23BA  0E 00                 MVI  C,H'00' SET COMMA POSITION CTR TO Ø (NO COMMA'S ALLOWED IN EXPONENTIAL
23BC  CC 65 24              CALL X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT
23BF  F1                    POP  PSW FETCH FLAG STATUS & LOST (GAINED) DIGIT COUNT FROM STACK
23C0  F4 2E 24              CP   X242E IF # IN ACC >Ø, ADD N TRAILING Ø'S, INSERT DECIMAL-PT WHERE
23C3  C1                    POP  B RESTORE BC FROM STACK - B IS LEFT NUMERIC COUNT, C IS TOTAL NUMERIC COUNT
23C4  F1                    POP  PSW FETCH FLAG STATUS FROM STACK
23C5  CC C7 1A              CZ   X1AC7 IF RIGHT NUMERIC COUNT IS Ø, REMOVE '.' CHAR IN BUFR BY DCX H (PUT IN
23C8  F1                    POP  PSW FETCH EXPONENT ADJUSTMENT & FLAG STATUS FROM STACK
23C9  DA CF 23              JC   X23CF JMP IF # TO CONVERT IS Ø - see 238B
23CC  83                    ADD  E  ADD DIGIT COUNT - 6 OR 16   } COMPUTE (N-6)+M-K
23CD  90                    SUB  B  SUBTRACT # OF DIGITS TO LEFT OF DP
23CE  92                    SUB  D  # IN D ACCOUNTS FOR SIGN CHAR
23CF  C5          X23CF     PUSH B PUT LEFT NUMERIC COUNT ON STACK (B), DON'T CARE (C)
23D0  CD 43 22              CALL X2243 CONVERT EXPONENT OF #, PUT CHAR'S IN BUFR
23D3  EB                    XCHG PUT ADDR OF NULL OF CONVERTED # STRING IN HL (see 2265)
23D4  D1                    POP  D PUT LEFT NUMERIC COUNT IN D
23D5  C3 8F 22              JMP  X228F CONVERSION ALMOST DONE - CLEAN UP STRING, LEADING ZERO SUPPRESSION, INSERT SIGN CHAR

23D8  D5          X23D8     PUSH D  SAVE DE ON STACK
23D9  AF                    XRA  A  CLEAR ACC - USED AS CTR (NEGATIVE COUNT FOR X1Ø, POSITIVE COUNT FOR ÷1Ø)
23DA  F5                    PUSH PSW SAVE EXPONENT-ADJUST CTR ON STACK
23DB  CD 0A 24              CALL X240A TEST IF # IS LESS THAN (10^6 SNG-PREC) or (1Ø^16 DBL-PREC) - RETURN HERE
23DE  F7          X23DE     RST  6  TEST TYPE OF VALUE IN HOLDING AREA
23DF  EA EE 23    DBL-PREC  JPE  X23EE JMP IF VALUE IS DBL-PREC TYPE
23E2  01 43 91              LXI  B,X9143 } LOAD BCDE WITH SNG-PREC CONSTANT = 1Ø^5
23E5  11 F9 4F              LXI  D,X4FF9 }
23E8  CD A1 19              CALL X19A1 PERFORM SNG-PREC MAGNITUDE COMPARISON
23EB  C3 F4 23              JMP  X23F4
23EE  11 31 25    X23EE     LXI  D,X2531 LOAD DE WITH ADDR OF DBL-PREC CONSTANT = 1Ø^15
23F1  CD 0C 18              CALL X180C PERFORM DBL-PREC MAGNITUDE COMPARISON
23F4  F2 07 24    X23F4     JP   X2407 JMP IF (SNG-PREC # ≥ 1Ø^5) or (DBL-PREC # ≥ 1Ø^16)
23F7  F1          EXIT      POP  PSW FETCH EXPONENT-ADJUST CTR FROM STACK
23F8  CD CA 20    X23F8     CALL X20CA MULTIPLY FLOATING-PT VALUE IN HOLDING AREA BY 1Ø - DECR ACC BY 1
23FB  F5                    PUSH PSW SAVE EXPONENT-ADJUST CTR ON STACK
23FC  C3 DE 23    X1Ø LOOP  JMP  X23DE LOOP UNTIL (SNG-PREC # ≥ 1Ø^5) or (DBL-PREC VALUE ≥ 1Ø^16)
23FF  F1          X23FF     POP  PSW FETCH EXPONENT-ADJUST CTR FROM STACK
2400  CD D7 20    X2400     CALL X20D7 DIVIDE FLOATING-PT VALUE IN HOLDING AREA BY 1Ø - INCR ACC BY 1
2403  F5                    PUSH PSW SAVE EXPONENT-ADJUST CTR ON STACK
2404  CD 0A 24    ÷1Ø LOOP  CALL X240A TEST IF (SNG-PREC # < 10^6) or (DBL-PREC # < 1Ø^16) - return here if it is, if not
2407  F1          X2407     POP  PSW FETCH EXPONENT-ADJUST CTR FROM STACK
2408  D1                    POP  D RESTORE DE FROM STACK
2409  C9                    RET

240A  F7          X240A     RST  6 TEST TYPE OF VALUE IN HOLDING AREA
240B  EA 1A 24              JPE  X241A JMP IF VALUE IS DBL-PREC TYPE
240E  01 74 94              LXI  B,X9474 } LOAD BCDE WITH SNG-PREC CONSTANT = 1Ø^6
2411  11 F8 23              LXI  D,X23F8 }
```

ON THE LO END OF ORIGINAL # BY MAKING THOSE DIGITS PART OF A FRACTION.
ADD 1 TO # IN ACC CONVERTED)
& SNG-PREC
FORMAT XX.XXE±XX
CONVERSION IS 6 OR 16 DIGITS
OF LEADING Ø'S TO SKIP BEFORE INSERTING
REASON - 1# THIS # IS THE # OF THE DP
(SIGN CHAR ALREADY COUNTED - 23AF)
BY CALL 2451 AT 252B, 2326, or 2432)
IF ORIGINAL # IS .XXXXXX · 1Ø^N
AFTER ADJUSTMENT # IS XXXXXX. · 1Ø^(N-6)
EXPONENT ADJUSTMENT IS N-6
DESIRED FORMAT TO PRINT
XX.XXXX
K DIGITS
M DIGITS
M=6 SNG-PREC
M=16 DBL-PREC

ADJUST FLOATING-PT # IN HOLDING AREA SUCH THAT
SNG-PREC # 1Ø^5 ≤ # < 1Ø^6
DBL-PREC # 1Ø^15 ≤ # < 1Ø^16
SNG-PREC TEST # ≥ 1Ø^5
DBL-PREC TEST # ≥ 1Ø^15

TEST IF
SNG-PREC # < 1Ø^6
DBL-PREC # < 1Ø^16
NORMAL RETURN IF IT IS
÷1Ø IF IT ISN'T - LOOP TILL IT
to 23FF to ÷1Ø til it is

```
2414  CD A1 19          CALL X19A1 PERFORM SNG-PREC MAGNITUDE COMPARISON - TEST IF VALUE IN HOLDING AREA IS
2417  C3 29 24          JMP  X2429 SKIP 2 INSTRUCTIONS
241A  11 39 25  X241A   LXI  D,X2539 LOAD DE WITH ADDR OF DBL-PREC CONSTANT = 10^16
241D  CD DC 19          CALL X19DC MOVE CONSTANT TO DBL-PREC TEMP STORAGE AREA - PERFORM. DBL-PREC MAGNITUDE COMPARISON
2420  E1        X2429   POP  H  PUT RETURN ADDR IN HL
2421  F2 FF 23          JP   X23FF IF # ≥ 10^6 (SNG-PREC) or # ≥ 10^16 (DBL-PREC) - DIVIDE BY 10 TIL < CONSTANT - COUNT # IN ACC
2424  F9                PCHL    RETURN IF # IS LESS THAN

2425  B7        X2425   ORA  A  TEST # IN ACC
2426  C8        X2426   RZ   RETURN IF DONE, POSITION CTR IS ZERO
2427  3D                DCR  A  DECREMENT POSITION CTR
2428  36 30             MVI  M,A'0' STORE A '0' IN BUFR
242A  23                INX  H  ADVANCE PTR TO NXT LOC IN BUFR.
242B  C3 26 24          JMP  X2426 LOOP UNTIL CTR IS 0

242E  C2 35 24  X242E   JNZ  X2435
2431  C8        X2431   RZ   RETURN IF DONE, POSITION CTR IS ZERO
2432  CD 51 24          CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY
2435  36 30     X2435   MVI  M,A'0' STORE A '0' IN BUFR
2437  23                INX  H  ADVANCE PTR TO NXT LOC IN BUFR
2438  3D                DCR  A  DECREMENT POSITION CTR
2439  C3 31 24          JMP  X2431 LOOP UNTIL CTR IS 0

243C  7B        X243C   MOV  A,E PUT EXPONENT-ADJUSTMENT IN ACC (see 2314)  (E is + #)
243D  82                ADD  D  ADD DIGIT COUNT  (see 23B8 '06 FOR SNG-PREC, see 22EE 10 FOR DBL-PREC)
243E  3C                INR  A  ADD 1 TO ACC (COMPENSATES FOR DCR B AT START OF LOOP AT 2451)
243F  47                MOV  B,A PUT VALUE FOR DECIMAL-PT POSITION CTR IN B (USED AT 2451)
2440  3C                INR  A  ADD 1 TO ACC
2441  D6 03     X2441   SUI  H'03'  } COMPUTE COMMA POSITION CTR VALUE USING MODULO 3 ARITHMETIC
2443  D2 41 24          JNC  X2441  }
2446  C6 05             ADI  H'05'  } PUT THIS VALUE IN C (USED AT 245D
2448  4F                MOV  C,A    }
2449  3A CD 03  X2449   LDA  XOBCD LOAD ACC WITH BIT-FLAG BYTE (see 22D2)
244C  E6 40             ANI  A'@' TEST 4Ø BIT (SET IF COMMA'S TO BE PUT TO LEFT OF DECIMAL-PT - see p. 88)
244E  C0                RNZ  RETURN ACC ≠Ø IF COMMA'S TO BE PUT IN CONVERTED #
244F  4F                MOV  C,A CLEAR C REG IF NO COMMA'S (C USED TO COUNT DIGIT POSITIONS
2450  C9                RET      FOR INSERTING COMMA'S)

2451  05        X2451   DCR  B     DECREMENT DECIMAL-PT CTR
2452  C2 50 24          JNZ  X2450. IF NOT INSERTING A DECIMAL-PT, TRY FOR A COMMA
2455  36 2E     X2455   MVI  M,A'.' PUT DECIMAL-PT IN CHAR BUFR
2457  22 05 03          SHLD X0305  SAVE ADDR OF DECIMAL-PT IN FLAG AREA
245A  23                INX  H     ADVANCE CHAR BUFR PTR
245B  48                MOV  C,B   CLEAR REG C, NO MORE COMMA'S CAN BE INSERTED
245C  C9                RET

245D  0D        X2450   DCR  C     DECREMENT COMMA CTR
245E  C0                RNZ        RETURN IF NOT READY TO INSERT A COMMA
245F  36 2C             MVI  M,A',' PUT COMMA IN CHAR BUFR
2461  23                INX  H     ADVANCE CHAR BUFR PTR
2462  0E 03             MVI  C,H'03' SET REG C TO COUNT 3 MORE DIGITS BEFORE NEXT COMMA
2464  C9                RET

2465  D5        X2465   PUSH D  SAVE CONTENTS OF DE ON STACK
2466  F7                RST  6  TEST VALUE TYPE OF #-TO-BE-CONVERTED
2467  E2 92 24          JPO  X2492 JMP IF VALUE IS SNG-PREC TYPE
246A  C5                PUSH B  SAVE DEC-PT + COMMA CTR'S ON STACK
246B  E5                PUSH H  SAVE PTR TO CHAR BUFR ON STACK
246C  CD 91 19          CALL X1991 PUT DBL-PREC VALUE TO BE CONVERTED IN TEMP STORAGE AREA Ø418 - Ø41F
246F  21 41 25          LXI  H,X2541 LOAD HL WITH ADDR OF DBL-PREC CONSTANT = .5
2472  CD 9B 1B          CALL X1B8B MOVE DBL-PREC CONSTANT TO HOLDING AREA  Ø4BE - Ø415
2475  CD 4A 1E          CALL X1E4A ADD .5 TO # TO BE CONVERTED USING DBL-PREC ADD ROUTINE
2478  AF                XRA  A  CLEAR CARRY BIT - SIGNIFIES SPECIAL RETURN (see 1D23)
2479  CD 02 10          CALL X1002 PERFORM DBL-PREC INT FCN ON # IN HOLDING AREA
247C  E1                POP  H  FETCH PTR TO CHAR BUFR FROM STACK
247D  C1                POP  B  FETCH DEC-PT & COMMA CTR'S FROM STACK
```

```
M!    5760  } RZ  # = CONSTANT
C $   5761  } RNC # > CONSTANT  > 10^6
9%    5762  } RC  # < CONSTANT
M\    5763  }
      5764
      5765  # OF ÷10 PERFORMED
      5766

7     5767  STORE N LEADING OR TRAILING Ø'S
H     5768  IN BUFR, N IS # IN ACC
=     5769
60    5770
//    5771
C&$   5772

B5$   5773  STORE N LEADING OR TRAILING Ø'S
H     5774  IN BUFR, N IS # IN ACC
MQ$   5775
60    5776
@     5777
=     5778
C1$   5779

      5780  COMPUTE POSITION-CTR VALUES FOR
      5781  DECIMAL-PT POSITION + COMMA POSITION
<     5782  (IF ',' FORMAT USED) IN CONVERTED-# STRING
G     5783
<     5784
V     5785
RA$   5786
F     5787
O     5788
:M    5789  ZERO COMMA POSITION CTR IF NOT USING
      5790  ',' FORMAT - INHIBITS COMMA'S AT 245D
@     5791
O     5792
I     5793

      5794  PRINT USING FORMATTER
B)$   5795  INSERT DECIMAL POINT
6.    5796  WHEN B HAS COUNTED TO Ø
"U    5797  CANCEL ALL COMMA'S AFTER '.'
#     5798
H     5799
I     5800

      5801  INSERT COMMA WHEN C
@     5802  HAS COUNTED TO Ø
6,    5803  INSERT COMMA EVERY 3RD
#     5804  DIGIT THEREAFTER
H     5805
I     5806

U     5107  CONVERT ADJUSTED MAGNITUDE
      5808  OF FLOATING-PT # TO CHAR
25    5809  STRING EQUIVALENT
E     5810
      5911
M@4IF 5812  - # HAS BEEN ADJUSTED SUCH THAT
!A%   5813    10^15 ≤ # < 10^16.
M     5814
MJ    5815  ROUNDOFF OF LO-ORDER DIGIT
/     5816
M     5917
      5818
A     5819
```

```
247E  11 51 25      LXI  D,X2551 LOAD DE WITH START ADDR OF FLOATING-POINT CONSTANT TABLE
2481  3E 0A         MVI  A,H'0A' SET DIGIT-CONVERT LOOP CTR TO 10 DECIMAL
2483  CD 51 24  X2483 CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY
2486  C5                PUSH B SAVE DEC-PT + COMMA DIGIT CTR'S ON STACK
2487  F5                PUSH PSW SAVE DIGIT-CONVERT LOOP CTR ON STACK
2488  E5                PUSH H  PUT PTR TO CHAR BUFR AREA ON STACK
2489  D5                PUSH D  PUT PTR TO FLOATING POINT CONSTANT TABLE ON STACK
248A  06 2F             MVI  B,A'/' PRESET REG B TO 1 LESS THAN '0'
248C  04          X248C INR  B  ADVANCE CHAR (ASCII CHAR IN ACC)
248D  E1                POP  H } PUT PTR TO FLOATING-POINT CONSTANT TABLE IN HL
248E  E5                PUSH H } LEAVE PTR ON STACK
248F  3E 9E             MVI  A,H'9E' } SUBTRACT 7 BYTE DECIMAL CONSTANT FROM # TO
2491  CD 1E 1F          CALL X1F1E   } CONVERT IN HOLDING AREA
2494  D2 8C 24          JNC  X248C IF # IN HOLDING IS STILL POSITIVE, PERFORM ANOTHER SUBTRACTION
2497  E1                POP  H  RESET HL TO POINT TO START OF DECIMAL CONSTANT
2498  3E 8E             MVI  A,H'8E' } ADD CONSTANT TO # IN HOLDING AREA - MAKES # IN
249A  CD 1E 1F          CALL X1F1E   } HOLDING AREA POSITIVE AGAIN
249D  EB                XCHG SAVE PTR TO FIRST BYTE OF NEXT DECIMAL CONSTANT IN DE
249E  E1                POP  H  FETCH PTR TO CHAR BUFR FROM STACK
249F  70                MOV  M,B STORE CHAR FROM CONVERTED # IN CHAR BUFR
24A0  23                INX  H  ADVANCE CHAR BUFR PTR
24A1  F1                POP  PSW FETCH DIGIT-CONVERT LOOP CTR FROM STACK
24A2  C1                POP  B  REMOVE DEC-PT + COMMA CTR'S FROM STACK
24A3  3D                DCR  A  DECREMENT LOOP CTR
24A4  C2 83 24          JNZ  X2483 IF NOT ZERO, JMP TO CONVERT ANOTHER DIGIT
24A7  C5                PUSH B SAVE DEC-PT + COMMA CTR'S ON STACK
24A8  E5                PUSH H SAVE PTR TO CHAR BUFR ON STACK
24A9  21 0E 04          LXI  H,X040E LOAD HL WITH ADDR OF FIRST BYTE OF # IN HOLDING AREA
24AC  CD 43 18          CALL X1843 MOVE 4 BYTES OF # FROM HOLDING AREA 040E TO HOLDING AREA 0412
24AF  C3 BE 24          JMP  X24BE PERFORM 3 BYTE CONSTANT, THEN 2 BYTE CONSTANT CONVERSIONS
24B2  C5          X24B2 PUSH B SAVE DEC-PT + COMMA CTR'S ON STACK
24B3  E5                PUSH H SAVE PTR TO CHAR BUFR ON STACK
24B4  CD 8D 18          CALL X188D ADD .5 TO SNG-PREC # IN HOLDING AREA  <-- NOT APPEAR TO AFFECT VALUE
24B7  3C                INR  A ADJUST EXPONENT (IN ACC FROM 188D) - SET ACC TO NON-ZERO VALUE - DOES
24B8  CD AF 1C          CALL X1CAF PERFORM SNG-PREC INT FCN ON # IN HOLDING AREA  <-- ROUNDOFF OF LO-ORDER DIGIT
24BB  CD 46 19          CALL X1946 STORE #-IN-BCDE IN HOLDING AREA 0412-0415
24BE  E1          X24BE POP  H REMOVE PTR TO CHAR BUFR FROM STACK
24BF  C1                POP  B REMOVE DEC-PT + COMMA CTR'S FROM STACK
24C0  AF                XRA  A CLEAR ACC + CARRY BIT
24C1  11 97 25          LXI  D,X2597 LOAD DE WITH START ADDR OF 3 BYTE DECIMAL CONSTANT TABLE
24C4  3F          X24C4 CMC  FIRST TIME SET CARRY, 2ND TIME CLEAR CARRY (2 3-BYTE CONVERSIONS ?)
24C5  CD 51 24          CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY
24C8  C5                PUSH B SAVE DEC-PT + COMMA DIGIT CTR'S ON STACK
24C9  F5                PUSH PSW SAVE CARRY FLAG STATUS ON STACK
24CA  E5                PUSH H  SAVE PTR TO CHAR BUFR ON STACK
24CB  D5                PUSH D PUT PTR TO DECIMAL CONSTANT TABLE ON STACK
24CC  CD 51 18          CALL X1851 LOAD #-TO-CONVERT INTO BCDE (B - DON'T CARE, CDE - INTEGER #)
24CF  E1                POP  H PUT PTR TO DECIMAL CONSTANT TABLE IN HL
24D0  06 2F             MVI  B,A'/' PRESET REG B TO 1 LESS THAN '0'
24D2  04          X24D2 INR  B  ADVANCE CHAR (ASCII CHAR IN B)
24D3  7B                MOV  A,E } PERFORM LO BYTE SUBTRACTION
24D4  96                SUB  M   }
24D5  5F                MOV  E,A }
24D6  23                INX  H ADVANCE PTR TO POINT TO MEDIUM BYTE OF DECIMAL CONSTANT
24D7  7A                MOV  A,D } PERFORM MED BYTE SUBTRACTION
24D8  9E                SBB  M   }
24D9  57                MOV  D,A }
24DA  23                INX  H ADVANCE PTR TO POINT TO HI BYTE OF DECIMAL CONSTANT
24DB  79                MOV  A,C } PERFORM HI BYTE SUBTRACTION
24DC  9E                SBB  M   }
```

```
Q%  5820
    5821
MQ$ 5822
E   5823
    5824
    5825
U   5826
/   5827
    5828
    5829
    5830
>   5831
M   5832
R $ 5833 SUBTRACTION
    5834
>   5835
M   5836
    5837
    5838
    5839
B   5840
    5841
A   5842
=   5843
B $ 5844
E   5845
    5846
    5847
MC  5848
C $ 5849
E   5850
    5851
M   5852
    5853
    5854
M   5855
    5856
A   5857
/   5858
%   5859
    5860
MQ$ 5861
E   5862
    5863 C D E
    5864 HI MED LO  <-- 3 REG'S HOLDING # TO BE CONVERTED TO ASCII STRING
U   5865
    5866
    5867
/   5868
    5869
    5870
    5871
    5872
    5873
    5874
    5875
W   5876
    5877
    5878
    5879
```

```
                                    } PART OF HI BYTE SUBTRACTION
24DD  4F          MOV  C,A
24DE  2B          DCX  H  } BACK-UP HL PTR TO POINT TO LO BYTE OF 3 BYTE DECIMAL
24DF  2B          DCX  H  }                                        CONSTANT
24E0  D2 D2 24    JNC  X2402 IF # IN C D E IS STILL POSITIVE, PERFORM ANOTHER SUBTRACTION
24E3  CC 43 19    CALL X1943 ADD CONSTANT TO # IN C D E (3 BYTE ADD) MAKES # IN
24E6  23          INX  H  ADVANCE PTR TO POINT TO LO BYTE OF NEXT DECIMAL CONSTANT
24E7  CC 46 10    CALL X1046 STORE 3 BYTE #-IN-C D E AT LOC  0414 0413 0412
24EA  EB          XCHG    SAVE PTR TO DECIMAL CONSTANT IN DE
24EB  E1          POP  H FETCH PTR TO CHAR BUFR FROM STACK
24EC  70          MOV  M,B STORE CHAR FROM CONVERTED # IN CHAR BUFR
24ED  23          INX  H ADVANCE CHAR BUFR PTR
24EE  F1          POP  PSW FETCH CARRY FLAG STATUS FROM STACK
24EF  C1          POP  B REMOVE DEC-PT + COMMA CTR'S FROM STACK
24F0  DA C4 24    JC   X24C4 IF CARRY IS SET (1ST TIME THRU) LOOP FOR 2ND PASS THRU
24F3  13          INX  D } ADVANCE PTR TO POINT TO LO BYTE OF 2 BYTE DECIMAL CONSTANT = 1000
24F4  13          INX  D } (SKIP LARGEST 2 BYTE DECIMAL CONSTANT)
24F5  3E 04       MVI  A,H'04' SET DIGIT-CONVERT LOOP CTR TO 4 (1000 100 10 1)
24F7  C3 00 25    JMP  X2500 JMP INTO 2 BYTE CONVERSION ROUTINE
-------------------------------------------------------------------------------------------
24FA  D5    X24FA PUSH D   SAVE CONTENTS OF DE ON STACK
24FB  11 90 25    LXI  D,X2590 PUT START ADDR OF DECIMAL CONSTANT TABLE IN DE
24FE  3E 05       MVI  A,H'05' PUT DIGIT-CONVERT LOOP COUNT IN ACC (2-BYTE CONSTANTS)
2500  CD 51 24 →X2500 CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY
2503  C5          PUSH B  SAVE DEC-PT & COMMA CTR'S ON STACK
2504  F5          PUSH PSW SAVE DIGIT-CONVERT LOOP CTR ON STACK
2505  E5          PUSH H. PUT PTR TO CHAR BUFR AREA ON STACK
2506  EB          XCHG  PUT PTR TO CONSTANT TABLE IN HL
2507  4E          MOV  C,M FETCH LO BYTE OF DECIMAL CONSTANT } PUT DECIMAL
2508  23          INX  H   ADVANCE PTR TO CONSTANT TABLE     } CONSTANT IN BC
2509  46          MOV  B,M FETCH HI BYTE OF DECIMAL CONSTANT }
250A  C5          PUSH B PUT DECIMAL CONSTANT ON STACK
250B  23          INX  H PTR POINTS TO NEXT DECIMAL CONSTANT IN TABLE
250C  E3          XTHL AFTER XTHL  HL = DECIMAL CONSTANT, PTR TO CONSTANT TABLE ON STACK
250D  EB          XCHG AFTER XCHG  DE = DECIMAL CONSTANT
250E  2A 12 04    LHLD X0412 FETCH # TO CONVERT FROM HOLDING AREA
2511  06 2F       MVI  B,A'/' PRESET REG B TO 1 LESS THAN '0'
2513  04    →X2513 INR  B   ADVANCE CHAR (ASCII CHAR IN B)
2514  7D          MOV  A,L   H      L      SUBTRACT DECIMAL CONSTANT IN
2515  93          SUB  E    -D-carry -E     DE FROM # IN HL. LOOP TIL
2516  6F          MOV  L,A   H      L      # IN HL IS SMALLER THAN CONSTANT
2517  7C          MOV  A,H
2518  9A          SBB  D
2519  67          MOV  H,A  THIS LOOP COUNTS THE # OF TIMES THE CONSTANT CAN
                              BE SUBTRACTED FROM THE # TO CONVERT. REG B IS THE CTR
251A  D2 13 25    JNC  X2513 IF # IN HL IS STILL POSITIVE, PERFORM ANOTHER SUBTRACTION
251D  19          DAD  D ADD CONSTANT TO # IN HL - MAKES # IN HL POSITIVE AGAIN
251E  22 12 04    SHLD X0412 STORE PARTIALLY CONVERTED # BACK INTO HOLDING AREA
2521  D1          POP  D PUT PTR TO DECIMAL CONSTANT TABLE IN DE
2522  E1          POP  H RESTORE PTR TO CHAR BUFR AREA FROM STACK
2523  70          MOV  M,B STORE CHAR FROM CONVERTED # IN CHAR BUFR
2524  23          INX  H ADVANCE CHAR BUFR PTR
2525  F1          POP  PSW RESTORE DIGIT-CONVERT CTR IN ACC
2526  C1          POP  B REMOVE DEC-PT & COMMA CTR'S FROM STACK
2527  3D          DCR  A DECREMENT LOOP CTR
2528  C2 00 25    JNZ  X2500 WHEN Ø, 5 ASCII DIGITS HAVE BEEN EXTRACTED FROM B
252B  CD 51 24    CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY
252E  77          MOV  M,A STORE A NULL AT END OF CHAR BUFR - (ACC NOT CHANGE IN 2451)
252F  D1          POP  D RESTORE DE TO ORIGINAL VALUE
2530  C9          RET  END OF CONVERSION
-------------------------------------------------------------------------------------------
2531  FD    X2531 DATA H'FD'  }
2532  FF          RST  7      } = 10^15 DBL-PREC CONSTANT
2533  9F          SBB  A      }
```

```
                                                                    5880
                                                                    5881
                                                                    5882
                                                                    5883
                                                                    5884 C D E POSITIVE AGAIN
                                                                    5885 CONSTANT STORED AT LOC IN HL
                                                                    5886
                                                                    5887 (B DON'T CARE)
                                                                    5888
                                                                    5889
                                                                    5890
                                                                    5891
                                                                    5892
                                                                    5893 LOOP
                                                                    5894
                                                                    5895
                                                                    5896
                                                                    5897
                                                                    5898 INTEGER CONVERSION
                                                                    5899
                                                                    5900 BINARY TO DECIMAL (ASCII)
                                                                    5901
                                                                    5902
                                                                    5903
                                                                    5904
                                                                    5905
                                                                    5906
                                                                    5907
                                                                    5908
                                                                    5909
                                                                    5910
                                                                    5911
                                                                    5912
                                                                    5913
                                                                    5914
                                                                    5915
                                                                    5916
                                                                    5917
                                                                    5918
                                                                    5919
                                                                    5920
                                                                    5921 (ASCII #)
                                                                    5922
                                                                    5923
                                                                    5924
                                                                    5925
                                                                    5926
                                                                    5927
                                                                    5928
                                                                    5929
                                                                    5930
                                                                    5931
                                                                    5932 # TO BE CONVERTED
                                                                    5933
                                                                    5934
                                                                    5935
                                                                    5936
                                                                    5937
                                                                    5938
                                                                    5939
```

```
2534  31 A  5F          LXI  SP,X5FA9   }
2537  63                MOV  H,E        }  = 10^15 DBL-PREC CONSTANT
2538  B2                ORA  D          }
2539  FE FF       X2539 CPI  H'FF'      }
253B  03                INX  B          }
253C  BF                CMP  A          }  = 10^16 DBL-PREC CONSTANT
253D  C9                RET             }
253E  18                DCX  D          }
253F  0E 86             MVI  C,H'86'    }
2541  00          X2541 NOP             }
2542  00                NOP             }  FLOATING POINT
2543  00                NOP             }  DOUBLE-PRECISION
2544  00                NOP             }  VARIABLE = .5
2545  00          X2545 NOP             }
2546  00                NOP             }  FLOATING POINT
2547  00                NOP             }  SINGLE-PRECISION
2548  80                ADD  B          }  VARIABLE = .5
2549  00          X2549 NOP
254A  00                NOP
254B  04                INR  B
254C  BF                CMP  A
254D  C9                RET
254E  18                DCX  D
254F  0E 86             MVI  C,H'86'
2551  00          X2551 NOP             }
2552  80                ADD  B          }
2553  C6 A4             ADI  H'A4'      }  CONSTANT = 1E15
2555  7E                MOV  A,M        }
2556  8D                ADC  L          }
2557  03                INX  B          }
2558  00                NOP             }
2559  40                MOV  B,B        }
255A  7A                MOV  A,D        }  CONSTANT = 1E14
255B  10                DATA H'10'      }
255C  F3                DI              }
255D  5A                MOV  E,D        }
255E  00                NOP             }
255F  00                NOP             }
2560  A0                ANA  B          }
2561  72                MOV  M,D        }  CONSTANT = 1E13
2562  4E                MOV  C,M        }
2563  18                DATA H'18'      }
2564  09                DAD  B          }
2565  00                NOP             }
2566  00                NOP             }
2567  10                DATA H'10'      }  CONSTANT = 1E12
2568  A5                ANA  L          }
2569  D4 E8 00          CNC  X00E8      }
256C  00                NOP             }
256D  00                NOP             }
256E  E8                RPE             }
256F  76                MOV  M,M        }  CONSTANT = 1E11
2570  48                MOV  C,B        }
2571  17                RAL             }
2572  00                NOP             }
2573  00                NOP             }
2574  00                NOP             }
2575  E4 0B 54          CPO  X540B      }  CONSTANT = 1E10
2578  02                STAX B          }
2579  00                NOP             }
```

```
1)_  5940
     5941
2    5942
     5943
     5944
?    5945
I    5946
     5947
6    5948
     5949
     5950
     5951
     5952
     5953
     5954
     5955
     5956
     5957
     5958
     5959
?    5960
I    5961
     5962
6    5963
     5964
     5965
F$   5966
     5967
     5968
     5969
     5970
@    5971
     5972
     5973
     5974
Z    5975
     5976
     5977
     5978
     5979
N    5980
     5981
     5982
     5983
     5984
     5985
%    5986
T    5987
     5988
     5989
     5990
     5991
H    5992
     5993
     5994
     5995
     5996
T    5997
     5998
     5999
```

```
257A  00              NOP                                                    6000
257B  00              NOP                                                    6001
257C  CA 9A 39        JZ    X399A     } CONSTANT = 1E9                 J ;  6002
257F  00              NOP                                                    6003
2580  00              NOP                                                    6004
2581  00              NOP                                                    6005
2582  00              NOP                                                    6006
2583  E1              POP   H         } CONSTANT = 1E8                       6007
2584  F5              PUSH  PSW                                              6008
2585  05              DCR   B                                                6009
2586  00              NOP                                                    6010
2587  00              NOP                                                    6011
2588  00              NOP                                                    6012
2589  80              ADD   B                                                6013
258A  96              SUB   M         } CONSTANT = 1E7                       6014
258B  98              SBB   B                                                6015
258C  00              NOP                                                    6016
258D  00              NOP                                                    6017
258E  00              NOP                                                    6018
258F  00              NOP                                                    6019
2590  40              MOV   B,B       } CONSTANT = 1,000,000 = 1E6      @   6020
2591  42              MOV   B,D                                        B     6021
2592  0F              RRC                                                    6022
2593  00              NOP                                                    6023
2594  00              NOP                                                    6024
2595  00              NOP                                                    6025
2596  00              NOP                                                    6026
2597  A0        X2597 ANA   B                                                6027
2598  86              ADD   M         3 BYTE CONSTANT = 100000               6028
2599  01 10 27        LXI   B,X2710   3 BYTE CONSTANT = 10000                6029
259C  00              NOP                                                    6030
259D  10        X259D DATA  H'10'     } LO BYTE                              6031
259E  27              DAA             } HI BYTE  INTEGER CONSTANT = 10000    6032
259F  E8              RPE                                                    6033
25A0  03              INX   B         INTEGER CONSTANT = 1000                6034
25A1  64              MOV   H,H                                              6035
25A2  00              NOP             INTEGER CONSTANT = 100                 6036
25A3  0A              LOAX  B                                                6037
25A4  00 INTEGER CONSTANT=1 NOP       INTEGER CONSTANT = 10                  6038
25A5  01 00 21 LXI H,1B13 LXI B,X2100 } LOAD HL WITH ADDR 1B13 - ROUTINE TO NEGATE FLOATING-PT  6039
25A8  13              INX   D         } # IN HOLDING AREA                    6040
25A9  1B              DCX   D                                                6041
25AA  E3              XTHL  } PUT RETURN ADDR IN HL, PUT ADDR 1B13 ON STACK  6042
25AB  E9              PCHL  } EXECUTE RETURN TO CALLING ROUTINE (CALL 25A7) BY PUTTING RETURN  6043  ADDR IN PC
25AC  CD 36 1B        CALL  X1B36 PUT SNG-PREC # IN HOLDING AREA ONTO STACK (ARG TO SQR)  M6  6044  SQR
25AF  21 45 25        LXI   H,X2545 LOAD HL WITH ADDR OF SNG-PREC CONSTANT = .5  !E%  6045
25B2  CD 43 1B        CALL  X1B43 PUT SNG-PREC CONSTANT = .5 IN HOLDING AREA  MC  6046
25B5  C3 BB 25        JMP   X25BB SKIP NXT INSTRUCTION                  C;%  6047
25B8  CD 45 1C  X25B8 CALL  X1C45 CALL CSGN ROUTINE - CONVERT VALUE IN HOLDING AREA TO SNG-PREC  ME  6048  ↓ y^X = EXP(X * LN Y)
25BB  C1        →X25BB POP   B  } PUT ARGUMENT TO SQR IN BCDE FROM STACK   A  6049
25BC  D1              POP   D  } also Y                                  Q  6050
25BD  EF              RST   5 FLOATING-PT SIGN TEST OF # IN HOLDING AREA     6051
25BE  CA F7 25  X=0 y^X=1 JZ  X25F7 IF EXPONENT (X in y^X) IS ZERO, EVALUATE e0=1  J %  6052
25C1  78              MOV   A,B  } TEST EXPONENT OF SNG-PREC # IN BCDE        6053
25C2  B7              ORA   A    }                                        7  6054
25C3  CA 03 19  ← DONE JZ   X1903 IF # IN BCDE IS 0, CHANGE # IN HOLDING AREA TO 0 -  JSQR  6055
25C6  D5        y=0   PUSH  D  } PUT SNG-PREC # IN BCDE (ARG TO SQR) ON STACK  U DONE  6056
25C7  C5        y^X=1 PUSH  B  }                                          E  6057
25C8  79              MOV   A,C  } TEST SIGN BIT OF ARG TO SQR                6058
25C9  F6 7F           ORI   H'7F'}                                           6059
```

```
25C8  CD 51 1B        CALL X1B51 MOVE SNG-PREC # IN HOLDING AREA INTO BCDE (also x)
25CE  F2 DF 25        JP   X25DF IF ARG IS POSITIVE, JMP
2501  D5              PUSH D  } PUT SNG-PREC # ON STACK (=.5 FOR SQR) (x for y^x)
2502  C5              PUSH B  }
2503  CD C5 1C        CALL X1CC5 PERFORM INT FUNCTION ON EXPONENT # (x in y^x)
2506  C1              POP  B  } PUT EXPONENT # IN BCDE FROM STACK
2507  D1              POP  D  }
2508  F5              PUSH PSW
2509  CD A1 1B        CALL X1BA1 TEST IF (EXPONENT #) = INT(EXPONENT #) - AFFECTS ZERO FLAG!!
250C  E1              POP  H
250D  7C              MOV  A,H
250E  1F              RAR
250F  E1        X25DF POP  H    } PUT ARG TO SQR IN HOLDING AREA FROM STACK
2510  22 14 04        SHLD X0414 } (also y in y^x)
2513  E1              POP  H     }
2514  22 12 04        SHLD X0412 }
2517  DC A7 25        CC   X25A7 IF EXPONENT IS AN ODD INTEGER, PUT ADDR 1B13 ON STACK TO NEGATE
251A  CC 13 1B        CZ   X1B13 IF Y IS MINUS & 2509 SHOWS EQUALITY, NEGATE Y TO POSITIVE #, OTHERWISE
251D  D5              PUSH D  } PUT EXPONENT # ON STACK (.5 FOR SQR, X in y^x)
251E  C5              PUSH B  }
251F  CD 98 19        CALL X1998 COMPUTE LN Y
2522  C1              POP  B  } FETCH EXPONENT # FROM STACK
2523  D1              POP  D  }
2524  CD 03 19        CALL X1903 COMPUTE PRODUCT X*LNY, THEN COMPUTE EXP 2 TO FINISH
2527  CD 36 19  X2527 CALL X1B36 PUT ARG (IN HOLDING AREA) ONTO STACK
252A  01 38 81        LXI  B,X8138 } LOAD BCDE WITH SNG-PREC CONSTANT = 1.4427 = 1/ln 2.
252D  11 3B AA        LXI  D,XAA3B }
2530  CD 03 19        CALL X1903 PERFORM SNG-PREC MULTIPLY, COMPUTE ARG*1.4427
2533  3A 15 04        LDA  X0415 LOAD ACC WITH EXPONENT BYTE OF PRODUCT
2536  FE 88           CPI  H'88' } IF EXPONENT IS 7. OR LARGER (2^7 = 127) JMP TO SIGN CHECK
2538  D2 C9 1A        JNC  X1AC9 } ARG MIGHT BE TOO LARGE TO COMPUTE e^x
253B  CD C5 1C        CALL X1CC5 PERFORM INT FUNCTION ON PRODUCT
253E  C6 80           ADI  H'80' } PERFORM ANOTHER MAGNITUDE TEST } MAY SET VALUE TO ZERO
2540  C6 02           ADI  H'02' } IF ARG IS TOO SMALL
2542  DA C9 1A        JC   X1AC9 }
2545  F5              PUSH PSW SAVE ADJUSTED EXPONENT
2546  21 87 19        LXI  H,X1987 LOAD HL WITH ADDR OF SNG-PREC CONSTANT = 1
2549  CD 90 18        CALL X1890 PERFORM SNG-PREC ADD, COMPUTE L9 = INT(1.4427*ARG)+1
254C  CD CD 19        CALL X19CD PERFORM SNG-PREC MULTIPLY, .693147 * L9
254F  F1              POP  PSW REMOVE ADJUSTED EXPONENT TO GET AT ARG ON STACK (see 25F7)
2550  C1              POP  B  } FETCH ARG FROM STACK INTO BCDE
2551  D1              POP  D  }
2552  F5              PUSH PSW PUT ADJUSTED EXPONENT BACK ONTO STACK
2553  CD 99 18        CALL X1899 PERFORM SNG-PREC SUBTRACT, E9 = -(.693147*L9 - ARG)
2556  CD 13 1B        CALL X1B13 INVERT SIGN OF RESULT, E9 = .693147*L9 - ARG
2559  21 37 26        LXI  H,X2637 LOAD HL WITH ADDR OF EXP POLYNOMIAL TABLE
255C  CD 67 26        CALL X2667 EVALUATE EXP POLYNOMIAL
255F  11 00 00        LXI  D,X0000 } PUT ADJUSTED EXPONENT INTO BCDE AS A SNG-PREC #
2562  C1              POP  B       } = 1.0 * 2^(ADJUSTED EXPONENT)
2563  4A              MOV  C,D     }
2564  C3 03 19        JMP  X1903 PERFORM SNG-PREC MULTIPLY - multiply polynomial by 2^K
2567  08        X2637 DATA H'08' CONSTANT TO INDICATE 8 SINGLE-PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE
2568  40              MOV  B,H     }
2569  2E 94           MVI  L,H'94' } SINGLE PRECISION CONSTANT
256B  74              MOV  M,H     } = -.00014131 6 ≃ 1/7076
256C  79              MOV  M,C     }
256D  4F              MOV  C,A     } SINGLE PRECISION CONSTANT
256E  2E 77           MVI  L,H'77' } = .00013298 8 ≃ 1/752
2570  6E              MOV  L,M     }
2571  02              STAX B       } SINGLE PRECISION CONSTANT
                                   } = -.00083013 6
```

```
MQ    6060
_%    6061
U     6062
E     6063
ME    6064
A     6065
Q     6066
      6067
      6068
      6069
      6070
      6071
      6072
..    6073
      6074
..    6075
\.%   6076 RESULT AT END OF EXP ROUTINE
      6077 LEAVE Y NEGATIVE TO CAUSE ERROR
U     6078 IN 1998 LOG ROUTINE
E     6079 FOR POSITIVE ARG 25C9 CLEARS ZERO
M     6080 FLAG
A     6081
Q     6082
MS    6083 SQR or y^x
M6    6084
8     6085  EXP
;*    6086
MS    6087  e^x = 2^K e^y
:     6088
      6089  where K = integer
RI    6090
ME    6091
F ?   6092
F .   6093
ZI    6094
      6095
!     6096
M     6097
MM    6098
      6099
A     6100
Q     6101
      6102
M     6103
M     6104
!7&   6105
M &   6106
      6107
A     6108
J     6109
CS    6110
      6111
2     6112
.     6113
      6114
      6115
0     6116
.     6117
      6118
      6119
```

```
2642 88                  ADC  B      } PART OF 3RD SINGLE PRECISION
2643 7A                  MOV  A,D    } CONSTANT = -.ØØ83Ø136 ≃ 1/12Ø
2644 E6 AØ               ANI  H'AØ'  } SINGLE PRECISION CONSTANT
2646 2A 7C 5Ø            LHLD X507C  }   = .Ø4165 74 = 1/24
2649 AA                  XRA  D      }
264A AA                  XRA  D      } SINGLE PRECISION CONSTANT
264B 7E                  MOV  A,M    }   = -.166665 = -1/6
264C FF                  RST  7      }
264D FF                  RST  7      } SINGLE PRECISION CONSTANT
264E 7F                  MOV  A,A    }
264F 7F                  MOV  A,A    }   = .5
2650 ØØ                  NOP         }
2651 ØØ                  NOP         } SINGLE PRECISION CONSTANT
2652 8Ø                  ADD  B      }
2653 81                  ADD  C      }   = -1
2654 ØØ                  NOP         }
2655 ØØ                  NOP         } SINGLE PRECISION CONSTANT
2656 ØØ                  NOP         }
2657 81                  ADD  C      }   = 1
2658 CD 36 13   X2658    CALL X1836  PUT ARG IN HOLDING AREA ONTO STACK
265B 11 99 1Ø            LXI  D,X1Ø39} PUT ADDR 1Ø89 ON STACK - MULTIPLY POLYNOMIAL BY ARG (SERIES IS AN ODD-TERM TYPE)
265E D5                  PUSH D      }
265F E5                  PUSH H SAVE PTR TO POLYNOMIAL TABLE ON STACK
2660 CD 51 18            CALL X1851 FETCH ARG IN HOLDING AREA INTO BCDE
2663 CD Ø3 19            CALL X1903 SNG-PREC MULTIPLY - COMPUTE (ARG * ARG)
2666 E1                  POP  H FETCH PTR TO POLYNOMIAL TABLE FROM STACK
2667 CD 36 18   X2667    CALL X1836 PUT # IN HOLDING AREA ONTO STACK
266A 7E                  MOV  A,M FETCH ITERATION CTR FROM TABLE (1ST BYTE IN A POLYNOMIAL TABLE)
266B 23                  INX  H ADVANCE PTR TO POLYNOMIAL TABLE
266C CD 43 18            CALL X1843 FETCH A SNG-PREC # FROM TABLE INTO HOLDING AREA
266F Ø6 F1      POP PSW  MVI  B,H'F1' DUMMY FETCH ITERATION CTR FROM STACK
2671 C1                  POP  B } FETCH # FROM STACK INTO BCDE
2672 D1                  POP  D }
2673 3D                  DCR  A DECREMENT ITERATION CTR
2674 C8         <DONE    RZ IF CTR IS ZERO, POLYNOMIAL HAS BEEN EVALUATED
2675 D5                  PUSH D } SAVE # IN BCDE ON STACK
2676 C5                  PUSH B }
2677 F5                  PUSH PSW SAVE ITERATION CTR ON STACK
2678 E5                  PUSH H SAVE PTR TO TABLE ON STACK
2679 CD Ø3 19            CALL X1903 PERFORM SNG-PREC MULTIPLY
267C E1                  POP  H FETCH PTR TO TABLE FROM STACK
267D CD 54 18            CALL X1854 FETCH A # FROM TABLE INTO BCDE
2680 E5                  PUSH H SAVE PTR TO STMT ON STACK
2681 CD 9C 18            CALL X189C PERFORM SNG-PREC ADD
2684 E1                  POP  H FETCH PTR TO TABLE FROM STACK
2685 C3 7Ø 26            JMP  X2670 LOOP UNTIL POLYNOMIAL HAS BEEN COMPLETELY EVALUATED
2688 EF                  RST  5 PERFORM FLOATING-PT SIGN TEST ON ARGUMENT VALUE (ALREADY IN HOLDING AREA)
2689 FA A5 26            JM   X26A5 IF ARG IS MINUS, JMP TO START A NEW SEQUENCE WITH ARG
268C 21 8A 26            LXI  H,X268A LOAD HL WITH ADDR OF LSB OF PREVIOUS RANDOM #
268F CD 43 18            CALL X1843 MOVE PREVIOUS RANDOM # INTO HOLDING AREA - FLAGS NOT AFFECTED
2692 C9                  RZ IF ARG WAS Ø, RESULT OF FCN EVALUATED IS THE PREVIOUS RANDOM #
2693 Ø1 35 9A            LXI  B,X9835} LOAD BCDE WITH SNG-PREC } MULTIPLY PREVIOUS RANDOM # BY
2696 11 7A 44            LXI  D,X447A} CONSTANT = 1.18795E+Ø7 }   1.18795E+Ø7
2699 CD Ø3 19            CALL X1903 PERFORM SNG-PREC MULTIPLY }
269C Ø1 28 68            LXI  B,X6828} LOAD BCDE WITH SNG-PREC } ADD 3.92768E-Ø8 TO ABOVE RESULT
269F 11 46 81            LXI  D,X3146} CONSTANT = 3.92768E-Ø8 }
26A2 CD 9C 18            CALL X189C PERFORM SNG-PREC ADD
26A5 CD 51 18   X26A5    CALL X1851 MOVE # IN HOLDING AREA INTO BCDE (EITHER PREVIOUS RANDOM #, OR NEW
26A8 7B                  MOV  A,E } SWAP MSB + LSB OF #
26A9 59                  MOV  E,C }
```

```
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
```

```
* P
*
*
M6
9
U
MQ
MS
M6
#
MC
A
Q
=
H
U
E
MS
MT
M
C &
%&
!:&
MC
H
5
D
MS
(
F1
M
MQ#)
Y
```

POLYNOMIAL EVALUATOR

TABLE OF COEFFICIENTS

ITERATION CTR - 1 BYTE

A_N } SNG-PREC CONSTANTS

A_{N-1} } SNG-PREC CONSTANTS

⋮ } SNG-PREC CONSTANTS

A_1 } SNG-PREC CONSTANTS

$A_\emptyset$ } SNG-PREC CONSTANTS

RESULT IN HOLDING AREA

$((A_N * X) + A_{N-1}) * X \ldots + A_\emptyset$

$A_N X^N + \ldots A_1 X + A_\emptyset$

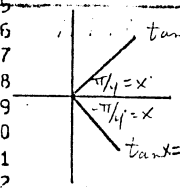
RND

- HL = Ø416

```
                                        } SWAP MSB & LSB OF #
26AA  4F              MOV   C,A
26AB  36 80           MVI   M,H'80'  SET SIGN BIT IN LOC #416 TO 1 (+ SIGN) see 1B73 FOR DERIVATION OF #416
26AD  2B              DCX   H  BACK UP PTR TO EXPONENT BYTE OF # IN HOLDING AREA
26AE  46              MOV   B,M  PUT EXPONENT IN B - REDUNDANT - see 1B51 AT 26A5
26AF  36 80           MVI   M,H'80'  SET EXPONENT TO 2^-1
2631  CD E0 18        CALL  X18E0  NORMALIZE # IN BCDE, STORE RESULT IN HOLDING AREA
2634  21 9A 26        LXI   H,X269A  LOAD HL WITH ADDR OF LSB OF WHERE RANDOM # IS TO BE SAVED
2637  C3 50 19        JMP   X1950  STORE RANDOM # BACK INTO RANDOM # HOLDING AREA FROM SNG-PREC

269A  52        X268A MOV   D,D  } SNG-PREC CONSTANT = .811635
269B  C7              RST   0
269C  4F              MOV   C,A  } RANDOM # HOLDING AREA
269D  80              ADD   B

269E  21 04 27  X268E LXI   H,X2704  LOAD HL WITH ADDR OF CONSTANT = π/2
26C1  CD 90 18        CALL  X1890  FETCH CONSTANT INTO BCDE, ADD TO ARGUMENT, THEN COMPUTE SIN

26C4  CD 36 13  X26C4 CALL  X1336  PUT ARG (#IN HOLDING AREA) ONTO STACK
26C7  01 49 83        LXI   B,X8349  } LOAD BCDE WITH SNG-PREC CONSTANT = 2π
26CA  11 DB 0F        LXI   D,X0FDB  }
26CD  CD 46 18        CALL  X1846  PUT CONSTANT IN HOLDING AREA
26D0  C1              POP   B  } FETCH ARG FROM STACK INTO BCDE
26D1  D1              POP   D  }
26D2  CD 2E 1A        CALL  X1A2E  PERFORM SNG-PREC DIVIDE, COMPUTE (ARG/2π)
26D5  CD 36 13        CALL  X1336  PUT RESULT OF DIVISION ONTO STACK
26D8  CD C5 1C        CALL  X1CC5  PERFORM INT FUNCTION ON RESULT OF DIVISION
26D9  C1              POP   B  } FETCH (ARG/2π) FROM STACK
26DC  D1              POP   D  }
26DD  CD 99 18        CALL  X1899  PERFORM SNG-PREC SUBTRACT, COMPUTE FRACTIONAL REMAINDER OF (ARG/2π)
26E0  21 08 27        LXI   H,X2708  LOAD HL WITH ADDR OF SNG-PREC CONSTANT = .25
26E3  CD 96 18        CALL  X1896  FETCH CONSTANT INTO BCDE, COMPUTE .25 - [ARG] WHERE -2π < ARG < 2π
26E6  EF              RST   5  PERFORM FLOATING-PT SIGN TEST ON RESULT
26E7  37              STC   SET CARRY TO SKIP CALL AT 26FB IF RESULT ≥ .25
26E8  F2 F0 26        JP    X26F0
26FB  CD 80 18        CALL  X1880  ADD .5 TO RESULT IN HOLDING AREA
26FE  EF              RST   5  PERFORM FLOATING-PT SIGN TEST ON NEW RESULT
26FF  B7              ORA   A  SET FLAGS - NC IF NEW RESULT ≥ 0
26F0  F5        X26F0 PUSH  PSW  SAVE FLAG STATUS ON STACK
26F1  F4 13 18        CP    X1813  IF POSITIVE, NEGATE FLOATING-PT # IN HOLDING AREA
26F4  21 08 27        LXI   H,X2708  LOAD HL WITH ADDR OF SNG-PREC CONSTANT = .25
26F7  CD 90 18        CALL  X1890  ADD CONSTANT = .25 TO RESULT IN HOLDING AREA
26FA  F1              POP   PSW  FETCH FLAG STATUS FROM STACK
26FB  D4 13 18        CNC   X1813  IF CARRY NOT SET, NEGATE # IN HOLDING AREA
26FE  21 0C 27        LXI   H,X270C  LOAD HL WITH ADDR OF SIN POLYNOMIAL TABLE
2701  C3 58 26        JMP   X2658  EVALUATE SIN POLYNOMIAL, RESULT LEFT IN HOLDING AREA

2704  08 0F     X2704 IN    H'0F'  } SINGLE PRECISION CONSTANT
2706  49              MOV   C,C  } = 1.5708 = π/2
2707  91              ADD   C

2708  00        X2708 NOP          } SINGLE PRECISION CONSTANT
2709  00              NOP          }
270A  00              NOP          } = .25
270B  7F              MOV   A,A    }

270C  05        X270C DCR   B  CONSTANT TO INDICATE 5 SINGLE PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE
270D  0A              CMP   D
270F  07              RST   2          = 39.7107
270F  1E 86           MVI   E,H'86'
2711  64              MOV   H,H
2712  26 99           MVI   H,H'99'    = -76.575
2714  87              ADD   A
2715  58              MOV   E,B
2716  34              INR   M
2717  23              INX   H          = 81.6022
2719  87              ADD   A
```

```
                    B C D E ← LSB BYTE
                      ↖ ACC
0     6180
0     6181  USE OF #416
4     6182
F     6183
6     6184
M     6185
!!&   6186
C}    6187  HOLDING AREA
R     6188
G     6189
O     6190
      6191
!'    6192  COS = SIN(ARG + π/2)
M     6193
M6    6194
I     6195  SIN
(     6196
MF    6197
A     6198
Q     6199
M.    6200
M6    6201
ME    6202
A.    6203
Q     6204
M     6205
!'    6206
M     6207
      6208
7     6209
  &   6210
M     6211
      6212
7     6213
      6214
      6215
!'    6216
M     6217
      6218
T     6219
!'    6220
CX&   6221
(     6222
I     6223
      6224
      6225
      6226
      6227
      6228
      6229
      6230
H     6231
      6232
      6233
&     6234
      6235
X     6236
4     6237
#     6238
      6239
```

```
2719 E0              RPO
271A 50              MOV  E,L     = -.411.3417                          )   6241
271B A5              ANA  L                                             %   6242
271C 86              ADD  M                                             '   6243
271D CA 0F 49        JC   X490F                                         Z I 6244
2720 83              ADD  E       = 6.28319 = 2π                            6245
2721 CD 36 19        CALL X1936 PUT ARG(IN HOLDING AREA) ONTO STACK     M6  6246
2724 CD C4 26        CALL X26C4 COMPUTE SIN(ARG) - RESULT IN HOLDING AREA MD& 6247   TAN
2727 C1              POP  B  } FETCH ARG FROM STACK, HALF OF # PUT IN HL DUE TO FORMAT A   6248
2728 E1              POP  H  } OF 1836 ROUTINE                                      6249
2729 CD 36 18        CALL X1836 PUT PARTIAL RESULT SIN(ARG) ON STACK    M6  6250
272C EB              XCHG PUT THE LO 2 BYTES OF ARG IN DE NOW - STANDARD FORMAT FOR SNG-PREC # 6251 IN BCDE
272D CD 46 18        CALL X1846 PUT # IN BCDE BACK INTO HOLDING AREA (ARG IN HOLDING AREA MF  6252
2730 CD 9E 26        CALL X269E COMPUTE COS(ARG) - RESULT IN HOLDING AREA M>& 6253
2733 C3 2C 1A        JMP  X1A2C COMPUTE SIN(ARG)/COS(ARG) - END OF TAN - RESULT IN HOLDING AREA C' 6254
2736 EF        X2736 RST  5  PERFORM SIGN TEST ON ARGUMENT TO ATN (IN HOLDING AREA)              6255
2737 FC A7 25        CM   X25A7 IF ARG IS MINUS PUT 1813 ON STACK TO INVERT RESULT WHEN DONE '% 6256
273A FC 13 18        CM   X1813 IF ARG IS MINUS, INVERT ARG (ARG FOR POLYNOMIAL MUST BE POSITIVE) 6257
273D 3A 15 04        LDA  X0415 TEST EXPONENT BYTE OF ARG IN HOLDING AREA :  6258
2740 FE 81           CPI  H'81' }                                           6259
2742 DA 51 27        JC   X2751 JMP IF ARGUMENT MAGNITUDE IS LESS THAN 1 ZQ' 6260
2745 01 00 81        LXI  B,X8100 } LOAD BCDE WITH SNG-PREC CONSTANT = 1    6261
2748 51              MOV  D,C     }                                         Q   6262
2749 59              MOV  E,C     }                                         Y   6263
274A CD 2E 1A        CALL X1A2E PERFORM SNG-PREC DIVIDE, COMPUTE 1/ARG  M.  6264
274D 21 96 18        LXI  H,X1896 } PUT 1896 ON STACK - WILL COMPUTE (π/2 - RESULT) see 2757 ! 6265
2750 F5              PUSH H       }                                         6266
2751 21 5B 27  X2751 LXI  H,X275B LOAD HL WITH ADDR OF ITERATION CTR OF ATN POLYNOMIAL TABLE !C' 6267
2754 CD 58 26        CALL X2658 EVALUATE ATN POLYNOMIAL                 MX& 6268
2757 21 04 27        LXI  H,X2704 LOAD HL WITH ADDR OF CONSTANT π/2 - used if 274D was executed ! ' 6269
275A C9              RET  END OF ATN - may do 1 or 2 other routines (see 2737 & 274D)  I   6270
275B 09        X275B DAD  B  CONSTANT TO INDICATE 9 SINGLE-PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE 6271
275C 4A              MOV  C,D } SINGLE PRECISION CONSTANT                   J   6272
275D D7              RST  2   }                                             W   6273
275E 38              DCX  SP  } = .00286623                                 ;   6274
275F 78              MOV  A,B }                                                 6275
2760 02              STAX B   } SINGLE PRECISION CONSTANT                       6276
2761 6E              MOV  L,M }                                                 6277
2762 84              ADD  H   } = -.0161657                                     6278
2763 78              MOV  A,E }                                                 6279
2764 FE C1           CPI  H'C1' } SINGLE PRECISION CONSTANT                 A   6280
2766 2F              CMA      }                                             /   6281
2767 7C              MOV  A,H } = .0429096                                      6282
2768 74              MOV  M,H } SINGLE PRECISION CONSTANT                       6283
2769 31 9A 7D        LXI  SP,X7D9A } = -.0752896                            1   6284
276C 84              ADD  H   } SINGLE PRECISION CONSTANT                       6285
276D 3D              DCR  A   }                                             =   6286
276E 5A              MOV  E,D } = .106563                                   Z   6287
276F 7D              MOV  A,L }                                                 6288
2770 C8              RZ       } SINGLE PRECISION CONSTANT                   H   6289
2771 7F              MOV  A,A }                                                 6290
2772 91              SUB  C   } = -.142089                                      6291
2773 7E              MOV  A,M }                                                 6292
2774 E4 B8 4C        CPO  X4CB8 } SINGLE PRECISION CONSTANT                 ;L  6293
2777 7E              MOV  A,M   } = .199936                                     6294
2778 6C              MOV  L,H } SINGLE PRECISION CONSTANT                       6295
2779 AA              XRA  D   }                                             *   6296
277A AA              XRA  D   } = -.333331                                  *   6297
277B 7F              MOV  A,A }                                                 6298
277C 00              NOP      } SINGLE PRECISION CONSTANT                       6299
                              } = 1
```



ATN

if x is minus, invert arg + invert result

if arg > 1, use 1/x as arg + use (π/2 - result) as result

```
                        NOP   } PART OF 9TH SINGLE
                        NOP   } PRECISION CONSTANT = 1
                        ADD C }
2780 00         X2780   NOP   ADDR OF 1ST BYTE AFTER BASIC
2781 00                 NOP
2782 00                 NOP
2783 00                 NOP
2784 00                 NOP
2785 00                 NOP
2786 00                 NOP
2787 00                 NOP
2788 00                 NOP
2789 00                 NOP
278A 21 44 29   X278A   LXI  H,X2944 LOAD HL WITH ADDR OF "CRLFLF WRITTEN BY ... "CHAR STRING
278D CD 41 12           CALL X1241  PRINT CHAR STRING ON OUTPUT CONSOLE
2790 21 F5 29   X2790   LXI  H,X29F5 } SET UP STACK AT 29F5
2793 F9                 SPHL        }
2794 22 00 03           SHLD X0300  STORE START ADDR OF STACK IN FLAG AREA (USED BY "NEW")
2797 DB 01              IN   H'01'  CLEAR TTY RECEIVE BUFR OF ANY RANDOM CHAR'S
2799 0E FF              MVI  C,H'FF' SET C TO -1 => DIFFERENCE BETWEEN INPUT CONSOLE PORT #
279B 11 FD 27           LXI  D,X27FD } PUT ADDR 27FD ON STACK FOR IMPLIED JMP
279E D5                 PUSH D      }
279F 3A FF 2F           LDA  X2FFF  FETCH DEFAULT CHANNEL #   } SENSE SWITCH SW8 IN UP
27A2 47                 MOV  B,A    SAVE # IN B               } POSITION MEANS DEFAULT
27A3 DB FF              IN   H'FF'  READ HI-8 SWITCHES ON FRONT PANEL } CHANNEL # WAS HAND -
27A5 1F                 RAR         ROTATE D8 INTO CARRY BIT      } LOADED INTO LOC 2FFF (12K)
                                                                      1FFF (8K).
27A6 DA B0 27           JC   X27B0  JMP IF SW8 WAS UP
27A9 E6 0C              ANI  H'0C'  MASK OUT ALL BUT BITS D11 + D12 (AFTER ROTATE, REMEMBER)
27AB CA B1 27           JZ   X27B1  IF SW8, SW11, + SW12 ALL IN DOWN POSITION, CHANNEL Ø J1'
27AE 06 10              MVI  B,H'10' IF SW8 DOWN, SW11 OR SW12 UP, CHANNEL 10H DEFAULT (4PIO
                                                                      or 2SIO)
27B0 78         X27B0   MOV  A,B    PUT CHANNEL # INTO ACC FROM B
27B1 32 FB 27   X27B1   STA  X27FB  STORE CHANNEL # OF STATUS PORT FOR LATER CHANGE OF2 '
27B4 DB FF              IN   H'FF'  READ HI-8 SWITCHES ON FRONT PANEL
27B6 17                 RAL         } TEST BIT CORRESPONDING TO SW14
27B7 17                 RAL         }
27B8 06 20              MVI  B,A' ; JZ
27BA 11 02 CA   X27BA   LXI  D,XCA02
27BD D8                 RC          JMP IF SW14 UP TO 27FD
27BE 17                 RAL         TEST BIT CORRESPONDING TO SW13
27BF 43                 MOV  B,E
27C0 1D                 DCR  E
27C1 D8                 RC          JMP IF SW13 UP TO 27FD
27C2 17                 RAL         TEST BIT CORRESPONDING TO SW12
27C3 DA DE 27           JC   X27DE  JMP IF SW12 UP
27C6 43                 MOV  B,E    JNZ
27C7 11 10 C2           LXI  D,XC210
27CA 17                 RAL         TEST BIT CORRESPONDING TO SW11
27CB D0                 RNC         JMP IF SW11 DOWN TO 27FD
27CC 17                 RAL         TEST BIT CORRESPONDING TO SW10
27CD 3E 03              MVI  A,H'03'
27CF CD FA 27           CALL X27FA
27D2 3C                 INR  A
27D3 3F                 CMC
27D4 87                 ADD  A
27D5 87                 ADD  A
27D6 3C                 INR  A
27D7 CD FA 27           CALL X27FA  A10 UP, SEND 1SH TO PORT 10H; A10 DOWN, SEND 11H TO PORT 10H
27DA 37                 STC         FORCE CARRY SET FOR IMPLIED JMP
27DB C3 BA 27           JMP  X27BA
27DE AF         X27DE   XRA  A      CLEAR ACC
```

SW14 UP → SIO A,B,C (REV Ø)
SW13 UP → 88-PIO
SW12 UP → 4PIO
SIO A,B,C (not REV Ø)
SW11 UP → 2SIO
4PIO

INITIALIZATION OF BASIC

+ OUTPUT CONSOLE STATUS PORT #
(IF INPUT + OUTPUT PORT THE SAME C = -1)

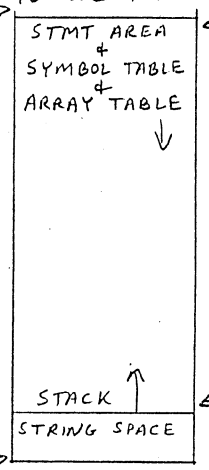
FOR SETTING UP THE IO DEVICE PORT STATUS POLARITY, PUT A CA (JZ) IN D (HI = READY), REG E HAS MASK FOR OUTPUT PORT, REG B HAS MASK FOR INPUT PORT

BASIC	STATUS BITS ACTIVE	I/O HEX MASKS
SIOA,B,C (REV Ø)	HI	2Ø/2
88PIO	HI	2/1
4PIO	HI	8Ø/8Ø
SIO A,B,C (not REV Ø)	LO	1/8Ø
2SIO	HI	1/2

```
270F  CD F4 27          CALL X27FA                                          M '   6360
27E2  CD F6 27          CALL X27F6                                          M '   6361
27E5  CD F6 27          CALL X27F6                                          M '   6362
27E8  4B                MOV  C,E                                            K     6363
27E9  2F                CMA                                                 /     6364
27EA  CD F6 27          CALL X27F6                                          M '   6365
27ED  3E 04             MVI  A,H'04'                                        >     6366
27EF  35                DCR  M                                              5     6367
27F0  CD FA 27          CALL X27FA                                          M '   6368
27F3  35                DCR  M                                              5     6369
27F4  35                DCR  M                                              5     6370
27F5  35                DCR  M                                              5     6371
27F6  21 F8 27   X27F6  LXI  H,X27F8  PUT ADDRESS OF I/O-STATUS-PORT-# IN HL ! '   6372
27F9  34                INR  M        INCREMENT TEMP STORAGE TO MAKE IT STATUS PORT # AGAIN 6373
27FA  D3 10      X27FA  OUT  H'10'    IF COMING FROM ABOVE, IGNORE, 27F8 USED TO SEND OUTPUT PORT INFO 6374 TO OUTPUT CONSOLE
27FC  C9                RET                                                 I     6375
------------------------------------------------------------------------------------------------------------------------------
27FD  62                X27FD MOV  H,D  PUT OP-CODE OF JMP-TIL-READY INSTRUCTION IN H (JNZ - LO, JZ - HI) 6376
27FE  68                MOV  L,B  PUT MASK-BYTE FOR INPUT PORT STATUS READY IN L      6377
27FF  22 EF 06          SHLD X06EF STORE THIS INFO IN ROUTINE THAT FETCHES A CHAR FROM " 6378 INPUT CONSOLE
2802  7C                MOV  A,H  } CHANGE H CONTENTS FROM A JMP TO A RETURN          6379
2803  E6 C8             ANI  H'C8'} (JNZ → RNZ LO, JZ → RZ HI)                  H     6380
2805  67                MOV  H,A  }  C2 → C0    CA → C8                       6381
2806  22 EE 07          SHLD X07EE STORE THIS INFO IN ROUTINE THAT            "     6382
2809  EE 0C             XRI  H'0C' } CHANGE H CONTENTS FROM A RETURN TO A CALL        6383
280B  67                MOV  H,A  } (RNZ → CZ   RZ → CNZ)                           6384
                                   C0 → CC   C8 → C4
280C  22 80 07          SHLD X0780 STORE THIS INFO IN CNTL-C CHK ROUTINE      "     6385
280F  EB                XCHG PUT OP CODE OF JMP-TIL-READY INSTR IN H, PUT MASK BYTE FOR OUTPUT PORT IN 6386 L
2810  22 E4 06          SHLD X06E4 STORE THIS INFO IN ROUTINE THAT PRINTS ASCII CHAR ON OUTPUT 6387 CONSOLE (RST 3)
2813  3A F8 27          LDA  X27F8 FETCH STATUS PORT # FROM TEMP STORAGE      : '   6388
2816  32 ED 06          STA  X06ED STORE STATUS PORT # IN ROUTINE TO INPUT ASCII CHAR 2  6389
2819  32 EC 07          STA  X07EC  "      "    "  " IN ROUTINE TO            2     6390
281C  32 7E 07          STA  X077E  "      "    "  " IN CNTL-C CHK ROUTINE    2     6391
281F  3C                INR  A    CONVERT BYTE IN ACC TO INPUT CONSOLE PORT # (ALWAYS 1 < 6392 MORE THAN STATUS PORT)
2820  32 F4 06          STA  X06F4 STORE PORT # IN ROUTINE TO INPUT ASCII CHAR 2      6393
2823  91                ADD  C    CONVERT BYTE IN ACC TO OUTPUT CONSOLE STATUS PORT #   6394
2824  32 E2 06          STA  X06E2 STORE OUTPUT CONSOLE STATUS PORT # IN ROUTINE TO PRINT 2 ASCII 6395 CHAR
2827  3C                INR  A    CONVERT BYTE IN ACC TO OUTPUT CONSOLE PORT #  <     6396
2828  32 EA 06          STA  X06EA STORE OUTPUT CONSOLE PORT # IN ROUTINE TO PRINT ASCII 2 CHAR 6397
282B  21 FF FF          LXI  H,X0000} SET PTR TO CURRENT-STMT-LINE # TO 0000          !     6398
282E  22 07 03          SHLD X0307  }                                         "W    6399
2831  AF                XRA  A    } CLEAR PRINT SUPPRESSION FLAG              /     6400
2832  32 A2 03          STA  X03A2 }                                          2"    6401
2835  CD 98 0A          CALL X0A98  PRINT CRLF + NULLS                        M     6402
2838  32 E9 03          STA  X03E9  CLEAR FLAG FOR AUTOMATIC EDIT WHEN SYNTAX ERROR 2 OCCURS 6403
283B  21 AA 03          LXI  H,X03AA} RESET PTR TO TABLE USED BY CHAR STRING PRINTING  !*    6404
283E  22 AB 03          SHLD X03AB }                                          "(    6405
------------------------------------------------------------------------------------------------------------------------------
2841  21 CB 29          LXI  H,X29CB LOAD HL WITH ADDR OF CHAR STRING "MEMORY SIZE"  !K)  6406 SET AMOUNT OF MEMORY
2844  CC 41 12          CALL X1241  PRINT CHAR STRING                         MA    6407
2847  CD E6 05          CALL X05E6  PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE, STOREM IN 6408 AVAILABLE TO BASIC
284A  D7                RST  2   SCAN OVER ANY LEADING "␣"'S IN LINE BUFR  LINE BUFR H     6409
284B  FE 41             CPI  A'A' } IF FIRST NON-SPACE CHAR IN LINE BUFR IS "A"  A     6410
284D  CA 8A 27          JZ   X278A } PRINT AUTHORS' NAMES AND REINITIALIZE     J '   6411
2850  B7                ORA  A    } IF FIRST NON-SPACE CHAR IS NOT A NULL OR COLON,  7     6412
2851  C2 68 28          JNZ  X2868} JMP TO ROUTINE TO CONVERT CHAR'S TO A #     B (   6413
2854  21 C7 29          LXI  H,X29D7 LOAD HL WITH ADDR OF 1ST BYTE AVAILABLE AFTER !W)  6414 BASIC
2857  23          X2857 INX  H      ADVANCE PTR TO NEXT BYTE OF MEMORY          #     6415
2858  3E 37             MVI  A,A'7' } STORE 37H IN MEMORY, THEN TEST TO SEE IF  >7    6416
285A  77                MOV  M,A    } THAT LOC IS UNPROTECTED RAM                     6417
285B  BE                CMP  M                                                >     6418
285C  C2 74 28          JNZ  X2874 JMP IF MEM LOC IS NOT FUNCTIONING AS A RAM LOC B ( 6419
```

```
285F 30          OCR  A   } STORE FF IN MEMORY LOC                    =    6420
2860 77          MOV  M,A }                                                6421
2861 BE          CMP  M   } TEST IF LOC. IS UNPROTECTED RAM            >    6422
2862 CA 57 28    JZ   X2857 TEST NEXT ADDRESS                          JW(  6423
2865 C3 74 28    JMP  X2874 JMP, END OF USABLE SEQUENTIAL RAM HAS BEEN FOUND  6424
2868 21 5A 03 X2868 LXI  H,X035A LOAD HL WITH ADDRESS OF LINE BUFR     !Z   6425
286B CD F3 08    CALL X08F3 ASSEMBLE GROUP OF DIGITS INTO 2-BYTE VALUE IN DE  6426  NOT TO EXCEED 65529
286E B7          ORA  A   } IF CHAR IN LINE BUFR AFTER LAST DIGIT IS NOT A NULL  6427
286F C2 49 04    JNZ  X049 } OR COLON, PRINT ERROR MESSAGE "SYNTAX ERROR" & RESTART  6428  INITIALIZATION
2872 EB          XCHG  PUT # FOR LAST ADDRESS AVAILABLE TO BASIC IN HL    6429
2873 2B          DCX  H } ADJUST #                                     +    6430
2874 2B       X2874 DCX  H }                                           +    6431
2875 E5          PUSH H PUT ADDR OF LAST BYTE AVAILABLE TO BASIC ON STACK  6432
-------------------------------------------------------------------------------------------------------------------------------------------
2876 21 7E 29 X2876 LXI  H,X297E LOAD HL WITH ADDR OF CHAR STRING "TERMINAL WIDTH"  !~  6433  SET TERMINAL WIDTH
2879 CD 41 12    CALL X1241 PRINT CHAR STRING                          MA   6434
287C CD E6 05    CALL X05E6 PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE INTO LINE  6435
287F D7          RST  2 SCAN OVER LEADING "␣"'S                        W BUFR  6436
2880 B7          ORA  A   } IF A RETURN WAS TYPED AFTER "?␣", LEAVE TERMINAL  7  6437
2881 CA AB 28    JZ   X28AB } SET TO 72 DECIMAL FOR TTY ('48H)'        J+(  6438
2884 21 5A 03    LXI  H,X035A LOAD HL WITH ADDR OF LINE BUFR           !Z   6439
2887 CD F3 08    CALL X08F3 ASSEMBLE GROUP OF DIGITS INTO A 2 BYTE VALUE  M   6440
288A 7A          MOV  A,D } TEST HI-ORDER BYTE OF VALUE                    6441
288B B7          ORA  A   } IF NOT Ø, START OVER, TERMINAL WIDTH MUST BE < 256  7  6442
288C C2 76 28    JNZ  X2876                                            8 (  6443
288F 7B          MOV  A,E } TEST LO-ORDER BYTE OF VALUE                    6444
2890 FE 10       CPI  H'10' }                                              6445
2892 DA 76 28    JC   X2876 IF < 16, START OVER AGAIN   16 ≤ WIDTH < 255  Z (  6446
2895 32 09 06    STA  X0609 STORE TERMINAL WIDTH IN ROUTINE TO PRINT ASCII CHAR  2Y  6447
2898 32 36 08    STA  X0836  "  "  "  "  "  NULL FUNCTION ROUTINE      26   6448
289B 32 91 0A    STA  X0A91  "  "  "  "  "  PRINT ROUTINE             2    6449
289E D6 0E    X289E SUI  H'0E' } COMPUTE WIDTH - [MOD14(WIDTH) + 14]    V    6450
28A0 D2 9E 28    JNC  X289E   }                                        R (  6451
28A3 C6 1C       ADI  H'1C'   } THIS # WILL IDENTIFY WHEN NEXT 14 CHAR F    6452
28A5 2F          CMA          } UNFORMATTED FIELD WILL NOT FIT ON A LINE /  6453
28A6 3C          INR  A       } (SEE ØABØ IN PRINT ROUTINE)            <    6454
28A7 83          ADD  E       }                                             6455
28A8 32 34 0A    STA  X0A34 STORE VALUE IN PRINT ROUTINE               24   6456
28AB 11 90 FF X28AB LXI  D,XFF90 LOAD DE WITH 16 BIT VALUE = -99 DECIMAL (AMOUNT OF  6457  SPACE FOR STRINGS - DEFAULT VALUE)
28AE E1          POP  H FETCH ADDR FROM STACK OF LAST USABLE BYTE FOR BASIC  6458
28AF 22 A6 03    SHLD X03A6 } SET START ADDR OF STRING SPACE TO HIGHEST AVAIL. ADDR  ¦&  6459
28B2 22 C8 03    SHLD X03C8 } SET END OF STRING SPACE TO SAME ADDR     "K   6460
28B5 19          DAD  D COMPUTE (LAST ADDR) - 100 = ADDR WHERE STRING SPACE ENDS  6461
28B6 D2 9E 04    JNC  X049E IF (LAST ADDR) WAS 98 OR LESS PRINT MESSAGE "OUT OF MEMORY"  6462  + RESTART INITIALIZATION
28B9 2B          DCX  H ADJUST ADDR TO 1 LESS THAN LAST STRING AREA ADDR  +    6463
28BA E5          PUSH H SAVE ADDR ON STACK                                 6464
-------------------------------------------------------------------------------------------------------------------------------------------
28BB 21 2F 29 X2800 LXI  H,X292F LOAD HL WITH ADDR OF CHAR STRING "WANT SIN-...-ATN?"  /)  6465  SAVE OR ELIMINATE TRIG
28BE CD 41 12    CALL X1241  PRINT CHAR STRING                         MA   6466
28C1 CD E6 05    CALL X05E6  PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE INTO  MLINE  6467  FUNCTIONS
28C4 D7          RST  2      SCAN OVER LEADING "␣"'S                   W BUFR  6468
28C5 FE 59       CPI  A'Y'   TEST 1ST NON-SPACE CHAR IN LINE BUFR      Y    6469
28C7 11 80 27    LXI  D,X2780 LOAD DE WITH ADDR OF BYTE AFTER ATN FUNCTION ROUTINE  6470
28CA CA F1 28    JZ   X28F1 JMP IF 1ST CHAR WAS A 'Y', RETAIN ALL TRIG FCNS  J (  6471
28CD FE 41       CPI  A'A' } IF 1ST CHAR WAS A 'A', DELETE ONLY THE ATN FUNCTION  A  6472
28CF CA 07 28    JZ   X2807 }                                          JW(  6473
28D2 FE 4E       CPI  A'N' } IF 1ST CHAR WAS NOT A 'N', THEN ANSWER TO QUESTION  N  6474
28D4 C2 B8 28    JNZ  X28B8 } IS WRONG. REPEAT QUESTION                B;(  6475
28D7 21 EE 08 X2807 LXI  H,X08EE LOAD HL WITH ADDR OF ROUTINE TO PRINT ERROR MESSAGE  !  6476  "ILLEGAL FUNCTION CALL"
28DA 11 36 27    LXI  D,X2736 LOAD DE WITH ADDR OF BYTE AFTER SIN-COS-TAN FUNCTIONS  6'  6477
28DD 22 57 00    SHLD X0057 DELETE REFERENCE TO ATN FUNCTION SUBROUTINE ADDR - CHANGE  "W  6478  ADDR TO ERROR MESSAGE ROUTINE
28E0 FE 41       CPI  A'A' TEST 1ST CHAR IN LINE BUFR AGAIN            A    6479
```

```
28E2  CA F1 28        JZ    X28F1 IF CHAR WAS A 'A', THEN ONLY ATN IS TO BE DELETED - JMP      J (  6480
28E5  22 51 00        SHLD  X0051 DELETE COS  } FUNCTION REFERENCES - CHANGE TO ERROR          "Q   6481
28E8  22 55 00        SHLD  X0055 DELETE TAN  } MESSAGE ROUTINE - "ILLEGAL FUNCTION CALL"      "U   6482
28EB  22 53 00        SHLD  X0053 DELETE SIN  }                                               "S   6483
28EE  11 BE 26        LXI   D,X26BE LOAD DE WITH ADDR OF BYTE AT END OF BASIC NOT INCLUDING>&  6484 TRIG FCNS (ADDR OF 1ST BYTE OF COS)
28F1  EB              XCHG  X28F1   PUT ADDR OF FIRST BYTE AFTER BASIC IN HL                  6485
28F2  36 00           MVI   M,H'00' STORE A NULL AT THAT LOC                          6       6486
28F4  23              INX   H       ADVANCE PTR                                       #       6487
28F5  22 0F 03        SHLD  X030F STORE THIS ADDR IN PTR THAT IDENTIFIERS THE 1ST BYTE AVAILABLE 6488 FOR STMT AREA
28F8  E3              XTHL  PUT ADDR OF STMT AREA ON STACK, PUT ADDR OF BYTE BELOW STRING 6489 SPACE IN HL
28F9  11 F5 29        LXI   D,X29F5 LOAD DE WITH ADDR 29F5 (ADDR WHERE STACK STARTS AT THE 6490 MOMENT)
28FC  E7              RST   4 TEST HL CONTENTS RELATIVE TO DE CONTENTS                       6491
28FD  DA 9E 04        JC    X049E IF HL < DE, STRING SPACE IS SOMEWHERE IN BASIC, PRINT "OUT OF 6492 MEMORY", RESTART INITIALIZATION
2900  D1              POP   D REMOVE ADDR OF STMT AREA FROM STACK                      Q       6493
2901  F9              SPHL  RESET SP TO START STACK HI IN MEMORY JUST BELOW STRING SPACE      6494
2902  22 0D 03        SHLD  X030D SAVE ADDR WHERE STACK STARTS IN LOC 030D            "}      6495
2905  EB              XCHG  PUT ADDR OF STMT AREA IN HL, PUT ADDR WHERE STACK STARTS IN DE    6496
2906  CD 94 04        CALL  X0494 TEST IF ENOUGH MEMORY AVAILABLE BETWEEN STMT AREA M*        6497 SP-46 (46 LOC FOR STACK MIN.)
2909  7B              MOV   A,E  }                                                            6498 END OF
290A  95              SUB   L    }                                                            6499 BASIC
290B  6F              MOV   L,A  } COMPUTE DIFFERENCE BETWEEN DE & HL                         6500
290C  7A              MOV   A,D  }                                                            6501
290D  9C              SBB   H    }   (DE) - (HL) = # OF BYTES AVAILABLE                       6502
290E  67              MOV   H,A  }                                                            6503
290F  01 F0 FF        LXI   B,XFFF0} SUBTRACT 16 DECIMAL FROM # IN HL                         6504
2912  09              DAD   B    }                                                            6505
2913  CD 98 0A        CALL  X0A98 PRINT CRLF & NULLS                                  M       6506
2916  CD 73 21        CALL  X2173 CONVERT # IN HL TO ASCII & PRINT IT                 M  !    6507
2919  21 90 29        LXI   H,X2990 LOAD HL WITH ADDR OF CHAR STRING "␣BYTES FREE...!"        6508
291C  CD 41 12        CALL  X1241  PRINT CHAR STRING                                  MA      6509
291F  21 41 12        LXI   H,X1241} AFTER INITIALIZATION IS COMPLETE, MODIFY COMMAND  !A      6510
2922  22 E6 04        SHLD  X04E6 } MODE PROCESSING ROUTINE TO CALL PRINT INSTEAD OF INIT.    6511
2925  CD 9A 05        CALL  X059A EXECUTE "NEW" - RESET ALL BASIC FLAGS & PTRS        M       6512
2928  21 0E 04        LXI   H,X040E} AT ADDR 0002, JMP TO COMMAND MODE PROCESSING     !^      6513
292B  22 02 00        SHLD  X0002 } ROUTINE, NOT INITIALIZATION                       "       6514 LAST BYTE OF RAM
292E  E9              PCHL  JMP TO ADDR 040E -- START BASIC EXECUTION                         6515
292F  57        X292F MOV   D,A                                                        W       6516 CHAR STRING
2930  41              MOV   B,C                                                       A       6517 "WANT SIN-COS-TAN-ATN"
2931  4E              MOV   C,M                                                       N       6518
2932  54              MOV   D,H                                                       T       6519 STRING TERMINATED BY A NULL
2933  20              DATA  A' '                                                              6520
2934  53              MOV   D,E                                                       S       6521
2935  49              MOV   C,C                                                       I       6522
2936  4E              MOV   C,M                                                       N       6523
2937  2D              DCR   L                                                         -       6524
2938  43              MOV   B,E                                                       C       6525
2939  4F              MOV   C,A                                                       O       6526
293A  53              MOV   D,E                                                       S       6527
293B  2D              DCR   L                                                         -       6528
293C  54              MOV   D,H                                                       T       6529
293D  41              MOV   B,C                                                       A       6530
293E  4E              MOV   C,M                                                       N       6531
293F  2D              DCR   L                                                         -       6532
2940  41              MOV   B,C                                                       A       6533
2941  54              MOV   D,H                                                       T       6534
2942  CE 00           ACI   H'00'                                                     N       6535
2944  0D        X2944 DCR   C                                                                 6536 CHAR STRING
2945  0A              LDAX  B                                                                 6537 CRLF LF "WRITTEN BY BILL
2946  0A              LDAX  B                                                                 6538 GATES &PAUL ALLEN &MONTE DAVIDOFF."
2947  57              MOV   D,A                                                       W       6539 CRLF
                                                                                                     STRING TERMINATED BY A NULL
```



```
2948  52              MOV  D,D
2949  49              MOV  C,C
294A  54              MOV  D,H
294B  54              MOV  D,H
294C  45              MOV  B,L
294D  4E              MOV  C,M
294E  20              DATA A' '
294F  42              MOV  B,D
2950  59              MOV  E,C
2951  20              DATA A' '
2952  42              MOV  B,D
2953  49              MOV  C,C
2954  4C              MOV  C,H
2955  4C              MOV  C,H
2956  20              DATA A' '
2957  47              MOV  B,A
2958  41              MOV  B,C
2959  54              MOV  D,H
295A  45              MOV  B,L
295B  53              MOV  D,E
295C  20              DATA A' '
295D  26 20           MVI  H,A' '
295F  50              MOV  D,B
2960  41              MOV  B,C
2961  55              MOV  D,L
2962  4C              MOV  C,H
2963  20              DATA A' '
2964  41              MOV  B,C
2965  4C              MOV  C,H
2966  4C              MOV  C,H
2967  45              MOV  B,L
2968  4E              MOV  C,M
2969  20              DATA A' '
296A  26 20           MVI  H,A' '
296C  4C              MOV  C,L
296D  4F              MOV  C,A
296E  4E              MOV  C,M
296F  54              MOV  D,H
2970  45              MOV  B,L
2971  20              DATA A' '
2972  44              MOV  B,H
2973  41              MOV  B,C
2974  56              MOV  D,M
2975  49              MOV  C,C
2976  44              MOV  B,H
2977  4F              MOV  C,A
2978  46              MOV  B,M
2979  46              MOV  B,M
297A  AE              XRA  M
297B  00              DCR  C
297C  0A              LDAX B
297D  00              NOP
297E  54        X297E MOV  D,H
297F  45              MOV  B,L
2980  52              MOV  D,D
2981  4C              MOV  C,L
2982  49              MOV  C,C
2983  4E              MOV  C,M
2984  41              MOV  B,C
2985  4C              MOV  C,H
```

```
R     6540
I     6541
T     6542
T     6543
E     6544
N     6545
      6546
B     6547
Y     6548
      6549
B     6550
I     6551
L     6552
L     6553
      6554
G     6555
A     6556
T     6557
E     6558
S     6559
      6560
&     6561
P     6562
A     6563
U     6564
L     6565
      6566
A     6567
L     6568
L     6569
E     6570
N     6571
      6572
&     6573
M     6574
O     6575
N     6576
T     6577
E     6578
      6579
D     6580
A     6581
V     6582
I     6583
D     6584
O     6585
F     6586
F     6587
.     6588
      6589
      6590
      6591
T     6592  CHAR STRING
E     6593
R     6594  "TERMINAL WIDTH"
M     6595
I     6596  STRING TERMINATED BY A
N     6597  NULL
A     6598
L     6599
```

```
2986  20              DATA A' '                                  6600
2987  57              MOV  D,A                            W      6601
2988  49              MOV  C,C                            I      6602
2989  44              MOV  B,H                            C      6603
298A  54              MOV  D,H                            T      6604
298B  C8              RZ                                  H      6605
298C  00              NOP                                        6606
-------------------------------------------------------------------------
298D  20        X298D DATA A' '                                  6607  CHAR STRING
298F  42              MOV  B,D                            B      6608
298F  59              MOV  E,C                            Y      6609  "␣ BYTES FREE" CRLF CRLF
2990  54              MOV  D,H                            T      6610
2991  45              MOV  B,L                            E      6611  "ALTAIR BASIC VERSION 3.2" CRLF
2992  53              MOV  D,E                            S      6612  "[EXTENDED VERSION]" CRLF
2993  20              DATA A' '                                  6613
2994  46              MOV  B,M                            F      6614
2995  52              MOV  D,D                            R      6615  STRING TERMINATED BY A NULL
2996  45              MOV  B,L                            E      6616
2997  C5              PUSH B                              E      6617
2998  0D              DCR  C                                     6618
2999  0A              LDAX B                                     6619
299A  0D              DCR  C                                     6620
299B  0A              LDAX B                                     6621
299C  41              MOV  B,C                            A      6622
299D  4C              MOV  C,H                            L      6623
299E  54              MOV  D,H                            T      6624
299F  41              MOV  B,C                            A      6625
29A0  49              MOV  C,C                            I      6626
29A1  52              MOV  D,D                            R      6627
29A2  20              DATA A' '                                  6628
29A3  42              MOV  B,D                            B      6629
29A4  41              MOV  B,C                            A      6630
29A5  53              MOV  D,E                            S      6631
29A6  49              MOV  C,C                            I      6632
29A7  43              MOV  B,E                            C      6633
29A8  20              DATA A' '                                  6634
29A9  56              MOV  D,M                            V      6635
29AA  45              MOV  B,L                            E      6636
29AB  52              MOV  D,D                            R      6637
29AC  53              MOV  D,E                            S      6638
29AD  49              MOV  C,C                            I      6639
29AE  4F              MOV  C,A                            O      6640
29AF  4E              MOV  C,M                            N      6641
29B0  20              DATA A' '                                  6642
29B1  33              INX  SP                             3      6643
29B2  2E 82           MVI  L,H'82'                        .2     6644
29B4  0C              DCR  C                                     6645
29B5  0A              LDAX B                                     6646
29B6  58              MOV  E,E                            [      6647
29B7  45              MOV  B,L                            E      6648
29B8  58              MOV  E,B                            X      6649
29B9  54              MOV  D,H                            T      6650
29BA  45              MOV  B,L                            E      6651
29BB  4E              MOV  C,M                            N      6652
29BC  44              MOV  B,H                            D      6653
29BD  45              MOV  B,L                            E      6654
29BE  44              MOV  B,H                            C      6655
29BF  20              DATA A' '                                  6656
29C0  56              MOV  D,M                            V      6657
29C1  45              MOV  B,L                            E      6658
29C2  52              MOV  D,D                            R      6659
```

```
29C3  53            MOV   D,E          S     6660
29C4  49            MOV   C,C          I     6661
29C5  4F            MOV   C,A          O     6662
29C6  4E            MOV   C,M          N     6663
29C7  00            DATA  H'00'        )     6664
29C8  0D            DCR   C                  6665
29C9  0A            LDAX  B                  6666
29CA  00            NOP                      6667
29CB  40      X29CB MOV   C,L          M     6668   CHAR STRING
29CC  45            MOV   B,L          E     6669   "MEMORY SIZE"
29CD  4D            MOV   C,L          M     6670
29CE  4F            MOV   C,A          O     6671   STRING TERMINATED BY A NULL
29CF  52            MOV   D,D          R     6672
29D0  59            MOV   E,C          Y     6673
29D1  20            DATA  A' '               6674
29D2  53            MOV   D,E          S     6675
29D3  49            MOV   C,C          I     6676
29D4  5A            MOV   E,D          Z     6677
29D5  C5            PUSH  B            E     6678
29D6  00            NOP                      6679
29D7  00      X29D7 NOP                      6680
29D8  00            NOP                      6681
29D9  00            NOP                      6682
29DA  00            NOP                      6683
29DB  00            NOP                      6684
29DC  00            NOP                      6685
29DD  00            NOP                      6686
29DE  00            NOP                      6687
29DF  00            NOP                      6688
```