```
0000  F3          X0000  DI
0001  C3 90 27     X0001  JMP  X2790
0004  11 1C 20     X0004  LXI  D,X201C
0007  1C                  INR  E
RST 1 0008  7E            MOV  A,M   PUT NXT CHAR FROM STMT AREA INTO ACC
0009  E3                  XTHL  SAVE PTR ON STACK, FETCH RETURN ADDR FROM STACK
000A  BE                  CMP  M  COMPARE WITH CHAR AT ADDR AFTER RST 1 SUBR CALL
000B  23          X0008   INX  H  ADVANCE HL TO RETURN TO ADDR AFTER ARGUMENT BYTE
000C  E3                  XTHL  SWAP PTR'S AGAIN
000D  C2 A9 04            JNZ  X04A9  SYNTAX ERROR MESSAGE
RST 2 0010  23            INX  H nxt char.   RZ  null or colon   } skip spaces
0011  7E                  MOV  A,M           RC  DIGIT
0012  FE 3A              CPI  A':'          RNC NOT A DIGIT
0014  D0                  RNC   RETURN NO CARRY, CHAR POINTED BY H,L IS ':' or larger char
0015  C3 06 07            JMP  X0706
RST 3 0018  F5            PUSH PSW   PUT CHAR TO PRINT ON STACK
0019  3A A2 03            LDA  X03A2 } LOAD ACC FROM PRINT SUPPRESSION FLAG + TEST FLAG
001C  B7                  ORA  A     }
001D  C3 CB 06            JMP  X06CB JMP TO OUTPUT CONSOLE BYTE-TRANSFER ROUTINE
RST 4 0020  7C            MOV  A,H   COMPARE HL + DE
0021  92                  SUB  D
0022  C0                  RNZ        RZ  HL = DE
0023  7D                  MOV  A,L   RC  HL < DE
0024  93                  SUB  E     RNC HL ≥ DE
0025  C9                  RET
0026  01 00 3A      X0026 LXI  B,X3A00  DBL-PREC STARTS AT ØY0E, SNG-PREC STARTS AT Ø412
RST 5 0029  79            LDA  Ø415  FETCH EXPONENT BYTE OF # IN HOLDING AREA (LOC Ø415)
002A  04                  INR  B
002B  B7                  ORA  A     TEST BYTE
002C  C2 EB 1A            JNZ  X1AEB
002F  C9                  RET        RZ : EXPONENT = ØØ   MAGNITUDE = Ø
RST 6 0030  3A A4 03      LDA  X03A4 FETCH TYPE CODE FROM FLAG AREA (INT=Ø2, SNG=Ø4, DBL=Ø8, CHAR=Ø3)
0033  FE 08               CPI  H'08' COMPARE TYPE CODE WITH DBL-PRECISION TYPE CODE
0035  C3 AE 0E            JMP  X0EAE CONTINUE THIS ROUTINE ELSEWHERE  RPO-SNG
RST 7 0038  C9            RET        USER FUNCTION AREA
0039  00                  NOP        SUBSTITUTE JMP INSTRUCTION TO USER PROGRAM
003A  00                  NOP
0038  18          X0038  DCX  D      SGN  BA  1818
003C  1B                  DCX  D
003D  BA                  CMP  D     INT  BB  1CBA
003E  1C                  INR  E
003F  08                  DATA H'08' ABS  BC  18Ø8
0040  1B                  DCX  D
0041  EE 08               XRI  H'08' USR  BD  Ø8EE
0043  B0                  ORA  B
0044  10                  DATA H'10' FRE  BE  1ØB0
0045  30                  DATA A'0'  INP  BF  143Ø
0046  14                  INR  D
0047  D1                  POP  D     POS  CØ  1ØD1
0048  10                  DATA H'10'
0049  AC                  XRA  H     SQR  C1  25AC
004A  25                  DCR  H
004B  88                  ADC  B     RND  C2  26FF
004C  26 98               MVI  H,H'98' LOG C3  1998
004E  19                  DAD  D
004F  F7                  RST  6     EXP  C4  25F7
0050  25                  DCR  H
0051  BE          X0051 CMP  H      COS  C5  26BE
0052  26 C4               MVI  H,H'C4' SIN C6  26C4
```

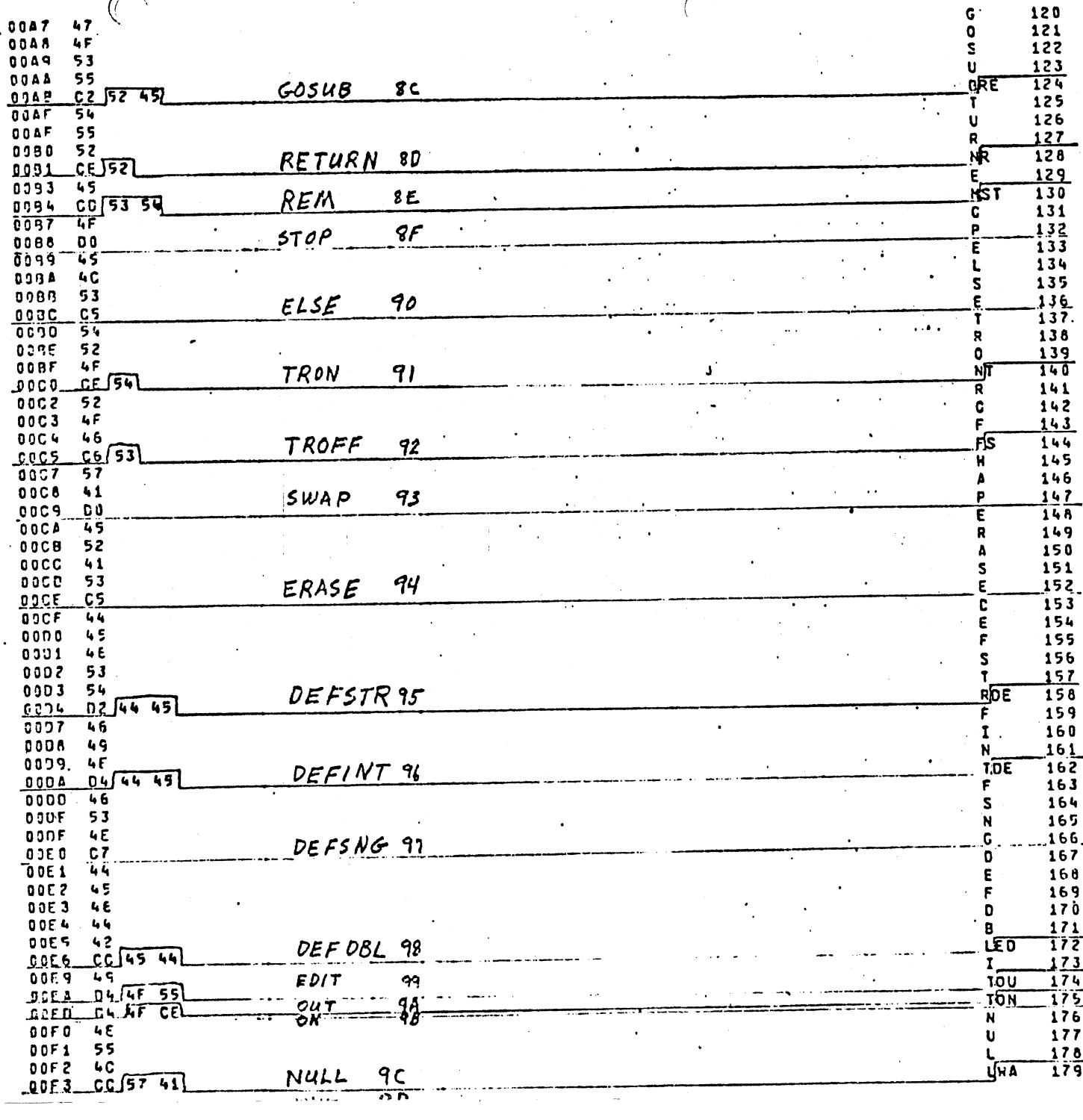
2E

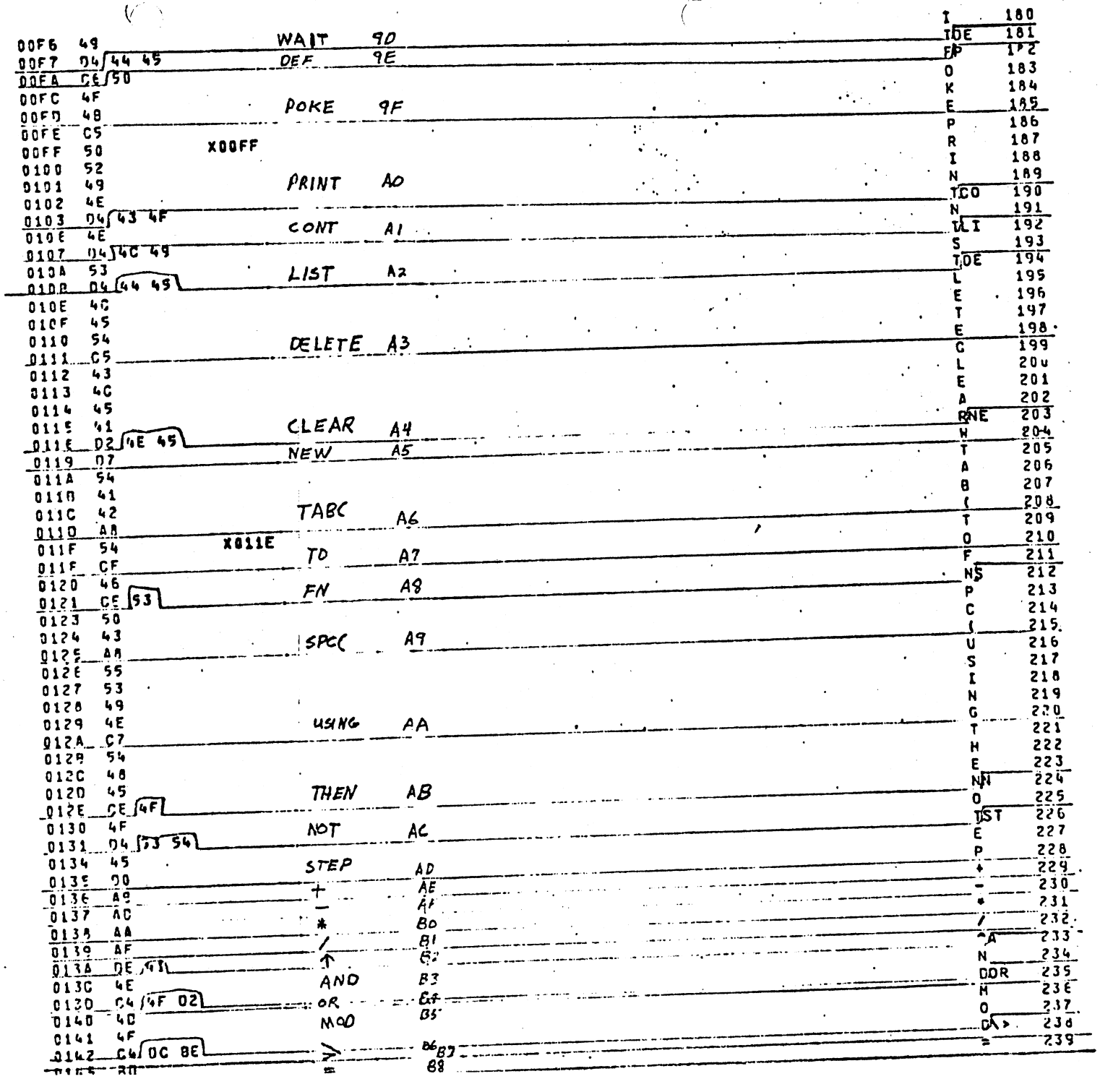
	FORMAT	PURPOSE
	RST 1	TEST BYTE POINTED TO
	ARG BYTE	BY HL FOR EQUALITY
		WITH BYTE AFTER RST 1
	RETURN WITH HL POINTING TO NXT	
	NON-SPACE BYTE	

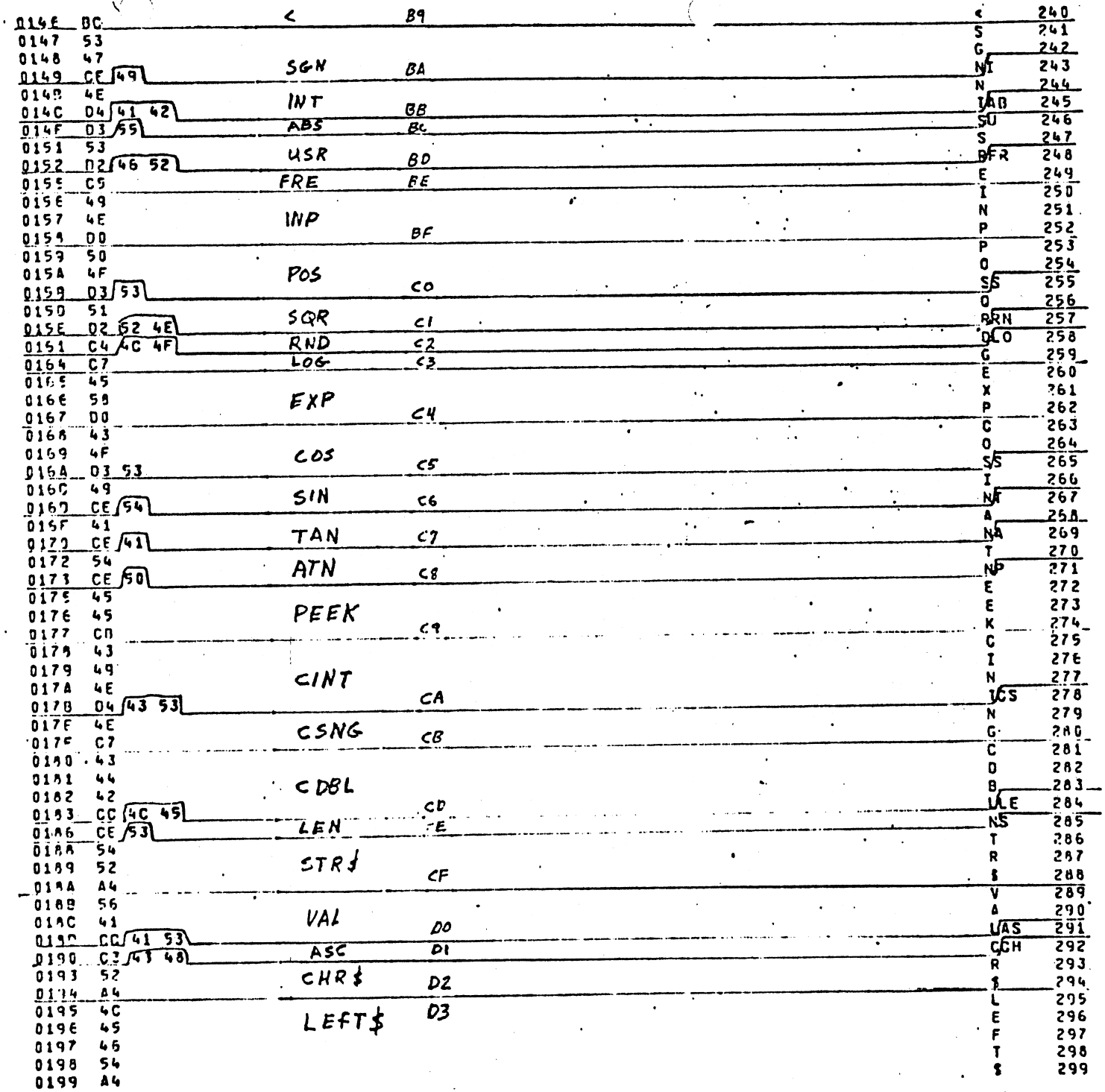
RST5 FLOATING POINT SIGN TEST
IF EXPONENT = Ø RZ
IF MSB OF MOST SIGNIFICANT FRACTION
BYTE IS ON RETURN WITH +1 IN ACC
AND SIGN BIT SET
IF MSB " " IS OFF SET ACC TO +1
WITH BIT OFF

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

```
       C4            √SIN C6        26C4                         ..
0054   28 12         √TAN C7        2721                         .
0056   27                                                        6.
0057   3E 27   X0057 √ATN C8        2736                         K
0059   4B            √PEEK C9       154B
005A   15            √CINT CA       1C11                          E
005B   11 1C 45
                     √C SGN CB      1C45
005E   1C
005F   6F            √CDBL CC       1C6F
0060   1C                                                        $
0061   A4            √LEN CD        13A4
0062   13            √STR$ CE       11CD                         M
0063   C0 11 8E      √VAL CF        148E
0066   14                                                        O
0067   B0            √ASC D0        13B0
0068   13                                                        >
0069   8E            √CHR$ D1       138E
006A   13                                                        N
006B   CE 13         √LEFT$ D2      13CE
006D   FF            √RIGHT$ D3     13FF
006E   13
006F   09            √MID$ D4       1409
0070   14
0071   79      X0071  +     RELATED TO THE ARITHMETIC OPERATORS
0072   79             -
0073   7C             *     ARITHMETIC OPERATOR PRECEDENCE TABLE
0074   7C             /
0075   7F             ↑                                          P
0076   50             AND                                        F
0077   46             OR
0078   7A             MOD
0079   7B      X0079  \
007A   45      X007A  E
007B   4E             N   D    80                      LAST LETTER OF EACH WORD
                                                       HAS MSB set (1)
007C   C4 46 4F       FOR      81
007F   D2 4E 45       NEXT     82
0082   58
0083   D4 44 41       DATA
0086   54                      83
0087   C1
0088   49             INPUT
0089   4E
008A   50                      84
008B   55
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    88
0099   54
009A   CF
009B   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
```


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 4F
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
RE 124
T 125
U 126
R 127
NR 128
E 129
ST 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 90
00F7 D4 44 45 DEF 9E
00FA C5 50
00FC 4F POKE 9F
00FD 4B
00FE C5
00FF 50 X00FF
0100 52
0101 49 PRINT A0
0102 4E
0103 D4 43 4F CONT A1
010E 4E
0107 D4 4C 49 LIST A2
010A 53
010B D4 44 45
010E 4C
010F 45
0110 54 DELETE A3
0111 C5
0112 43
0113 4C
0114 45
0115 41 CLEAR A4
011E D2 4E 45 NEW A5
0119 D7
011A 54
0110 41
011C 42 TABC A6
011D A8
011F 54 X011E
011F CF TO A7
0120 46 FN A8
0121 CE 53
0123 50
0124 43 SPCC A9
0125 A8
012E 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 53 54
0134 45 STEP AD
0135 D0 + AE
0136 A9 - AF
0137 AC * B0
0138 AA / B1
0139 AF ↑ B2
013A DE 4F
013C 4E AND B3
013D C4 4F 02 OR B4
0140 4D MOD B5
0141 4F
0142 C4 0C BE > B6 B7
= B8
I 180
TDE 181
FP 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
I 215
U 216
S 217
I 218
N 219
G 220
T 221
H 222
E 223
NN 224
O 225
UST 226
E 227
P 228
+ 229
- 230
* 231
/ 232
↑A 233
N 234
DDR 235
H 236
O 237
D(238
= 239


0146 BC < B9
0147 53
0148 47
0149 CE 49 SGN BA
014A 4E
014C D4 41 42 INT BB
014F D3 53 ABS BC
0151 53
0152 D2 46 52 USR BD
0155 C5 FRE BE
0156 49
0157 4E INP
0159 D0 BF
0159 50
015A 4F POS
0159 D3 53 C0
0150 51
015E D2 52 4E SQR C1
0161 C4 4C 4F RND C2
0164 C7 LOG C3
0165 45
016E 58 EXP
0167 D0 C4
0168 43
0169 4F COS
016A D3 53 C5
016C 49
016D CE 54 SIN C6
016F 41
0170 CE 41 TAN C7
0172 54
0173 CE 50 ATN C8
0175 45
017E 45 PEEK
0177 CB C9
0178 43
0179 49
017A 4E CINT
017B D4 43 53 CA
017F 4E
017F C7 CSNG CB
0180 43
0181 44
0182 42 CDBL
0183 CC 4C 45 CD
0186 CE 53 LEN CE
0188 54
0189 52 STR$
018A A4 CF
018B 56
018C 41 VAL
018D CC 41 53 D0
0190 C3 41 48 ASC D1
0193 52 CHR$ D2
0194 A4
0195 4C LEFT$ D3
019E 45
0197 46
0198 54
0199 A4
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
27E
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	40		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							314
01A9	07		√ NEXT	82	ØBD7		M	315
01AA	08		√ DATA	83	Ø98C			316
01AB	8C							317
01AC	09							318
01AD	0F		INPUT	84	ØBØF			319
01AE	08		√ DIM	85	ØEDB			320
01AF	08 0E		READ	86	ØB36		(	321
0101	3E 08		LET	87	Ø9AB		6	322
0103	AB						+	323
0104	09		√ GOTO	88	Ø95A			324
0105	5A						2	325
0106	09		√ RUN	89	Ø93D			326
0107	3D						=	327
0108	09		√ IF	8A	ØA16			328
0109	16 0A		√ RESTORE	8B	Ø7E1			329
010B	E1							330
010C	07		√ GOSUB	8C	Ø949		I	331
010D	49							332
010E	09		√ RETURN	8D	Ø976			333
010F	76							334
01C0	09		√ REM	8E	Ø98E			335
01C1	8E							336
01C2	09		√ STOP	8F	Ø7F5			337
01C3	F5							338
01C4	07		√ ELSE	90	Ø98E			339
01C5	8E							340
01C6	09		√ TRON	91	Ø83E		>	341
01C7	3E 08		√ TROFF	92	Ø83F		?	342
01C9	3F							343
01CA	08		√ SWAP	93	Ø844			344
01CB	44						0	345
01CC	08		√ ERASE	94	Ø87A			346
01CD	7A							347
01CE	08		√ DEF STR	95	Ø8A8			348
01CF	A8							349
01D0	08		√ DEF INT	96	Ø8AB		A	350
01D1	AB							351
01D2	08		√ DEF SGN	97	Ø8AE			352
01D3	AE							353
01D4	08		√ DEF DBL	98	Ø8B1			354
01D5	B1						1	355
01D6	08		√ EDIT	99	170C			356
01D7	0C							357
01D8	17		√ OUT	9A	143B			358
01D9	3B						;	359

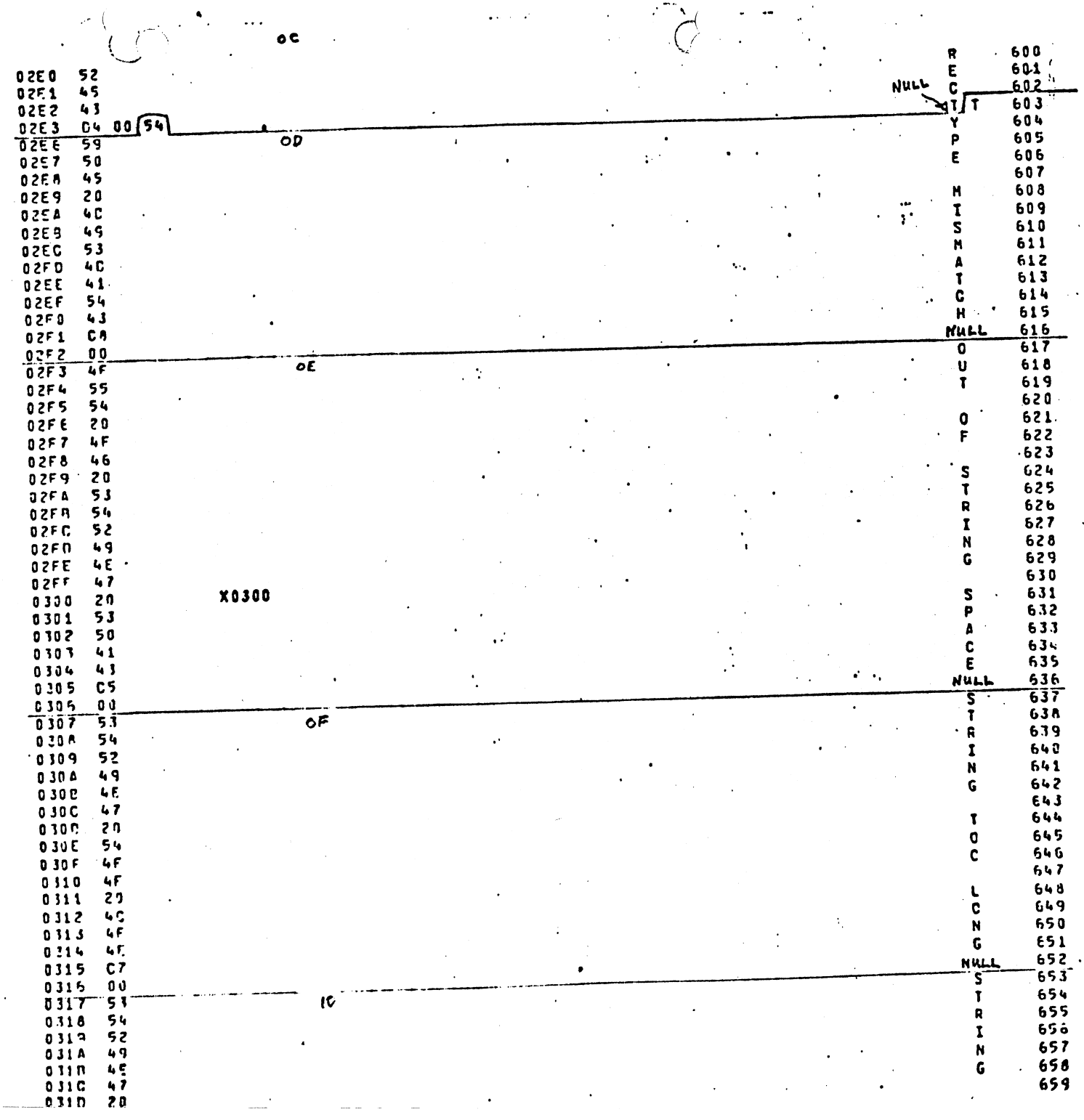
END OF TABLE

```
01DA  14                    ✓ OUT     9A    143B
01DB  FA 09 (30)            ✓ ON      9B    Ø9FA                                   O
01DE  03                    ✓ NULL    9C    Ø83Ø
01DF  41                                                                          A
01E0  14                    ✓ WAIT    9D    1441
01E1  0A 10 (52)              DEF     9E    1ØDA                                  Z R
01E4  15                    ✓ POKE    9F    1552
01E5  4C                                                                          L
01E6  04                    ✓ PRINT   A0    ØA4C
01E7  1D                    ✓ CONT    A1    Ø810
01E8  08                                                                          (
01E9  A8                    ✓ LIST    A2    14A8
01EA  14
01EB  1B                    ✓ DELETE  A3    151B
01EC  15
01ED  14                    ✓ CLEAR   A4    Ø914
01EE  09
01EF  99                    ✓ NEW     A5    Ø599
01F0  05
01F1  6F        X01F1       ✓ TABC    A6    1C6F = CDBL
01F2  1C
01F3  00                    ✓ TO      A7    ØØØØ - ADDR WHERE BASIC ORG'S
01F4  00
01F5  11 1C (88)            ✓ FN      A8    1C11 = CINT
01F8  1C                    ✓ SPCC    A9    1C88 = STRING VALUE CHECK             E
01F9  45                    ✓ USING   AA    1C45 = CSGN
01FA  1C                                                                          J
01FB  4A        X01FB       ✓ THEN    AB    1E4A = DBL-PREC ADDITION              C
01FC  1E (43)               ✓ NOT     AC    1E43 = DBL-PREC SUBTRACTION
01FF  1E (83)               ✓ STEP    AD    1F83 = DBL-PREC MULTIPLY
0200  1F                                                                          I
0201  C9                    ✓ +       AE    1FC9 = DBL-PREC DIVIDE
0202  1F
0203  0A                    ✓ -       AF    1CØA = DBL-PREC # MAGNITUDE COMPARISON (SIGN TEST FOR RESULT OF SUBTRACTION)
0204  1C
0205  9C        X0205       ✓ *       B0    189C = SNG-PREC ADDITION
0206  18
0207  99                    ✓ /       B1    1899 = SNG-PREC SUBTRACTION
020A  19
0209  03 19                 ✓ 4       B2    19D3 = SNG-PREC MULTIPLY              S
020B  2E 1A                 ✓ AND     B3    1A2E = SNG-PREC DIVIDE                .
020D  A1                    ✓ OR      B4    1BA1 = SNG-PREC # MAGNITUDE COMPARISON (SIGN TEST FOR RESULT OF SUBTRACTIO
020E  1B                                                                          U
020F  55        X020F       ✓ MOD     B5    1D55 = INTEGER ADDITION
0210  1D                                                                          I
0211  49                    ✓ \       B6    1D49 = INTEGER SUBTRACTION
0212  1D
0213  75                    ✓ >       B7    1D75 = INTEGER MULTIPLY
0214  1D                                                                          <
0215  BC                    ✓ =       B8    ØDBC = INTEGER DIVIDE
0216  00                                                                 NULL
0217  CC 1B 00              ✓ <       B9    1BCC = INTEGER # MAGNITUDE COMPARISON  L ← (SIGN TEST FOR RESULT OF SUBTRACTION)
021A  4E                      01            LAST CHAR IN EACH ERROR MESSAGE   N   ERROR MESSAGES
021B  45                                    HAS MSB SET (1)                   E
021C  5A                                                                      X
021D  54                                                                      T
021E  20        X021E
021F  57                                                                      W
0220  49                                                                      I
0221  54                                                                      T
```

Address	Bytes		Char	Line No.
0222	48		N	420
0223	4F	01	O	421
0224	55		U	422
0225	54		T	423
0226	20			424
0227	46		F	425
0228	4F		O	426
0229	02 00 53		R (NULL) S	427
022C	59	02	Y	428
022D	4E		N	429
022E	54		T	430
022F	41		A	431
0230	58		X	432
0231	20			433
0232	45		E	434
0233	52		R	435
0234	52		R	436
0235	4F		O	437
0236	02 00 52		(NULL) R	438
0239	45	03	E	439
023A	54		T	440
023B	55		U	441
023C	52		R	442
023D	4E		N	443
023E	20			444
023F	57		W	445
0240	49		I	446
0241	54		T	447
0242	48		H	448
0243	4F		O	449
0244	55		U	450
0245	54		T	451
0246	20			452
0247	47		G	453
0248	4F		O	454
0249	53		S	455
024A	55		U	456
024B	C2 00 4F		B (NULL) O	457
024E	55	04	U	458
024F	54		T	459
0250	20			460
0251	4F		O	461
0252	46		F	462
0253	20			463
0254	44		D	464
0255	41		A	465
0256	54		T	466
0257	C1		A	467
0258	00		NULL	468
0259	49	05	I	469
025A	4C		L	470
025B	4C		L	471
025C	45		E	472
025D	47		G	473
025E	41		A	474
025F	4C		L	475
0260	20			476
0261	46		F	477
0262	55		U	478
0263	4E		N	479

0264	43		05	C	480
0265	54			T	481
0266	49			I	482
0267	4F			O	483
0268	4E			N	484
0269	20				485
026A	43			C	486
026B	41			A	487
026C	4C			L	488
026D	CC	00 4F		NULL → L 0	489
0270	56		06	V	490
0271	45			E	491
0272	52			R	492
0273	46			F	493
0274	4C			L	494
0275	4F			O	495
0276	D7			W	496
0277	00			NULL	497
0278	4F		07	O	498
0279	55			U	499
027A	54			T	500
027B	20				501
027C	4F			O	502
027D	46			F	503
027E	20				504
027F	4D			M	505
0280	45			E	506
0281	4D			M	507
0282	4F			O	508
0283	52			R	509
0284	D9			Y	510
0285	00			NULL	511
0286	55		08	U	512
0287	4E			N	513
0288	44			D	514
0289	45			E	515
028A	46			F	516
028B	49			I	517
028C	4E			N	518
028D	45			E	519
028E	44			D	520
028F	20				521
0290	53			S	522
0291	54			T	523
0292	41			A	524
0293	54			T	525
0294	45			E	526
0295	4D			M	527
0296	45			E	528
0297	4E			N	529
0298	D4	00 53		NULL → T S	530
029B	55		09	U	531
029C	42			B	532
029D	53			S	533
029E	43			C	534
029F	52			R	535
02A0	49			I	536
02A1	50			P	537
02A2	54			T	538
02A3	20				539

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	D9		Y	571
02C4	00		NULL	572
02C5	44	0B	D	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		D	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 C8
02F2 00
0E
02F3 4F
02F4 55
02F5 54
02F6 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 4F
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
031F  4F                                 O
0320  52                                 R
0321  4D                                 M
0322  55                                 U
0323  4C                                 L
0324  41                                 A
0325  20
0326  54                                 T
0327  4F                                 O
0328  4F                                 O
0329  20
032A  43                                 C
032B  4F                                 O
032C  4D                                 M
032D  50                                 P
032E  4C                                 L
032F  45                                 E
0330  D8                                 X
0331  00                              NULL
```

11

```
0332  43                                 C
0333  41                                 A
0334  4E                                 N
0335  27                                 '
0336  54                                 T
0337  20
0338  43                                 C
0339  4F                                 O
033A  4E                                 N
0336  54                                 T
033C  49                                 I
033D  4E                                 N
033E  55                                 U
033F  C5                                 E
0340  00                              NULL
```

12

```
0341  55                                 U
0342  4E                                 N
0343  44                                 D
0344  45                                 E
0345  46                                 F
0346  49                                 I
0347  4E                                 N
0348  45                                 E
0349  44                                 D
034A  20
034B  55                                 U
034C  53                                 S
034D  45                                 E
034E  52                                 R
034F  20
0350  46                                 F
0351  55                                 U
0352  4E                                 N
0353  43                                 C
0354  54                                 T
0355  49                                 I
0356  4F                                 O
0357  CE 00                              N
                                      NULL
2C  0359  2C     DB   X0359  FLAG   USED BY INPUT FOR TEST AT 8849
59  035A  08     1. X035A  YOP  FIRST BYTE IN BUFFER . (LINE BUFFER - USED FOR INPUT & OUTPUT) NULL
```

```
0358  00          2.  NOP
0359  00              NOP
035A  00              NOP
035B  00              NOP
035C  00              NOP
035D  00              NOP
035E  00              NOP
035F  00              NOP
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
```

LINE BUFFER AREA

USED FOR INPUT + OUTPUT

```
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
```

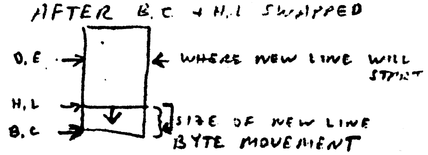
```
0397  00                62. NOP                                                    780
0398  00                    NOP                                                    781
0399  00                    NOP        LINE BUFFER AREA                            782
039A  00                    NOP                                                    783
039B  00                    NOP        USED FOR INPUT & OUTPUT                     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 (0D2E 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        3 BYTES PER ENTRY IN TABLE                  800
03AC  00                    NOP          CHAR COUNT - 1 BYTE                       801
03AD  00                    NOP          PTR TO 1ST BYTE OF CHAR STRING - 2 BYTES  802
03AE  00                    NOP                                                    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                                                    810
03B6  00                    NOP        03A8 CONTAINS 'STACK PTR' VALUE             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 - 1203 + 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          X03D1     NOP "NEW" SETS THIS LOC TO 03  ERASE SETS THIS LOC TO 1 FOR - 64  838
0302  00          X03D2     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
0304 00                 NOP  } ADDR OF STMT AREA                   } EXECUTED BY BASIC          841 } & NEXT
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 04A6)                         } 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 0000 - USED BY SOME INSTR - NOT USABLE AS DIRECT COMMAND  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 PTRS
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 02 (SYNTAX ERROR) TRIGGERS CALL  862 TO EDIT ROUTINE
03EA 00          X03EA  NOP  THIS AREA CHANGED TO 04 BY "NEW"                                 863
03EB 00                 NOP                                                                  864
03EC 00                 NOP    26 LOC - CORRESPONDS TO LETTERS 'A-Z'                         865
03ED 00                 NOP    SEE 0F21                                                      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 04 - THIS IS THE                         873
03F5 00                 NOP  "SIZE" OF SINGLE PRECISION VARIABLES                            874
03F6 00                 NOP                                                                  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 0  TRACE ON IF AF                      897
040D 00          X040D  NOP  LOST-BIT BYTE FOR DOUBLE-PREC RIGHT SHIFT (see 1F44) FOR 04BE-0415  898
040E 00          X040E  NOP  # HOLDING AREA, LSB FOR DOUBLE PRECISION VALUES                  899
```

HOLDING AREA

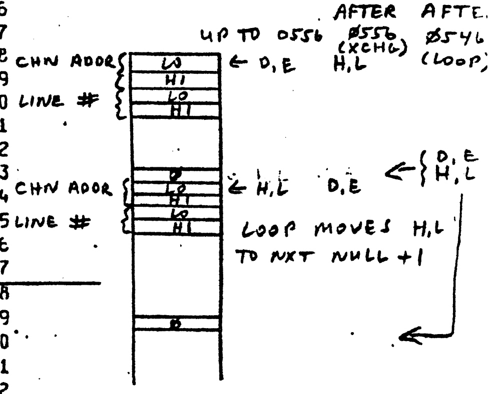
```
040F  00             NOP
0410  00    X0410    NOP                                       } DOUBLE
0411  00    X0411    NOP                                       } PRECISION
0412  00    X0412    NOP } INTEGER } SINGLE                    } VALUE
0413  00             NOP } VALUE   } PRECISION
0414  00    X0414    NOP           } VALUE
0415  00    X0415    NOP           }          Ø415 = EXPONENT BYTE FOR SINGLE OR DOUBLE PREC FLOATING POINT #'S
0416  00    X0416    NOP STORES SIGN BIT OF RESULT (see 1873)
0417  00    X0417    NOP LOST-BIT BYTE FOR Ø418-Ø41F
0418  00    X0418    NOP TEMP STORAGE FOR A DOUBLE PREC # (see 1891)
0419  00             NOP
041A  00             NOP
041B  00             NOP
041C  00             NOP
041D  00             NOP
041E  00    X041E    NOP
041F  00    X041F    NOP
0420  00    X0420    NOP UNDER SOME SPECIAL CASES, IF OVERFLOW OF FORMATTED OUTPUT FIELD, PUT '%' CHAR HERE
0421  00    X0421    NOP    BUFR AREA FOR # CONVERSION ROUTINE 2181
0422  00             NOP
0423  00    X0423    NOP      Ø421 IS USUALLY 1ST CHAR OF BUFR
0424  00             NOP
0425  00             NOP      Ø42Ø COULD BE 1ST CHAR IF OVERFLOW OF FORMATTED OUTPUT
0426  00             NOP                  (see 2200 or 22F9).
0427  00             NOP
0428  00             NOP
0429  00             NOP
042A  00             NOP
042B  00             NOP
042C  00             NCP
042D  00             NOP
042E  00             NOP
042F  00             NOP
0430  00             NOP
0431  00             NOP
0432  00             NOP
0433  00             NOP
0434  00             NOP
0435  00             NOP
0436  00             NOP
0437  00             NOP
0438  00             NOP
0439  00             NOP
043A  00             NOP
043B  00    X043B    NOP }  TEMP STORAGE FOR DOUBLE-PRECISION #
043C  00             NOP }   HOLDS MULTIPLIER  FOR DBL-PREC MULTIPLY
043D  00             NOP }   HOLDS DIVIDEND  FOR DBL-PREC DIVIDE
043E  00    X043E    NOP }
043F  00             NOP }
0440  00             NOP }
0441  00    X0441    NOP }
0442  00    X0442    NOP }
0443  20    X0443    DATA A' '                  space    CHAR STRING
0444  49             MOV  C,C                   I          WINU
0445  4E             MOV  C,M                   N
0446  40             ANA  B                     space
0447  00    X0447    NOP                        null
0448  0D    X0448    DCR  C                     CR       CHAR STRING
0449  0A             LDAX B                     LF       CR LF OK CR LF
044A  4F             MOV  C,A                   O
```

```
044B  CB                DATA H'CB'                                                    K
044C  00                DCR  C                                                        CR
044D  0A                LDAX B                                                        LF
044E  00                NOP                                                           NULL
-------------------------------------------------------------------------------------------------
044F  00       X044F    DCR  C                                                        CR    CHAR STRING
0450  0A                LDAX B                                                        LF    CR LF BREAK
0451  42                MOV  B,D                                                      B
0452  52                MOV  D,D                                                      R
0453  45                MOV  B,L                                                      E
0454  41                MOV  B,C                                                      A
0455  CB                DATA H'CB'                                                    K
0456  00                NOP                                                           NULL
-------------------------------------------------------------------------------------------------
0457  21 04 00 X0457    LXI  H,X0004 } IF GOSUB or FOR GROUP IS ON STACK, COMPUTE ADDR IN HL     STACK            GOSUB
045A  39                DAD  SP      } OF THE GOSUB or FOR TOKEN                      9     PTR TO GOSUB INSTRUCTION
045B  7E      ->X045B   MOV  A,M     FETCH BYTE FROM STACK AREA (POSSIBLY 'GOSUB' OR 'FOR' TOKEN)  LINE # OF GOSUB
045C  23                INX  H       ADVANCE HL TO PT TO LINE # OF GOSUB STORED ON STACK          GOSUB TOKEN   <-SP+4
                                                                    or FOR
045D  FE 81             CPI  H'81'  } RETURN IF BYTE IS NOT                                    CNTL-C CHK ADDR
045F  C0  <-TOKEN or byte on stack  RNZ } A 'FOR' TOKEN (PROCESSING A 'RETURN')        @    RETURN ADDR FOR CALL 0457 <- SP
            not a 'FOR' TOKEN
0460  4E                MOV  C,M  } PUT PTR TO VARIABLE VALUE REGION IN BC            N
0461  23                INX  H    } (MOST RECENT ACTIVE 'FOR' VARIABLE)               #
0462  46                MOV  B,M  }                                                   F
0463  23                INX  H                                                        #
0464  E5                PUSH H       SAVE PTR TO SIGN-OF-STEP ON STACK
0465  69                MOV  L,C } PUT PTR TO VARIABLE VALUE REGION IN HL (VARIABLE IN MOST RECENT ACTIVATED
0466  60                MOV  H,B }                                  'FOR' INSTRUCTION)
0467  7A                MOV  A,D } TEST IF DE IS 0000 - DE IS 0000 ONLY IF PROCESSING A 'NEXT' INSTRUCTION
0468  B3                ORA  E   } DE IS PTR TO VARIABLE VALUE REGION (see 09F7) OF VARIABLE IN 'FOR' STMT BEING
                                                                                      PROCESSED
0469  EB                XCHG     PUT PTR TO VAR IN CURRENT INSTR IN HL, PUT PTR TO VAR (FROM STACK) IN DE
046A  CA 6F 04          JZ   X046F JMP IF PROCESSING A 'NEXT' INSTRUCTION             J
046D  EB                XCHG     PUT PTR TO VAR (FROM STACK) IN HL, PUT PTR TO VAR IN CURRENT INSTR IN DE
046E  E7                RST  4   TEST IF GROUP OF BYTES ON STACK IS THE SAME VARIABLE AS CURRENT 'FOR' INSTR
046F  01 0E 00 ->X046F  LXI  B,X000E LOAD BC WITH 14 DECIMAL (# TO FIND EARLIER 'FOR' TOKEN ON STACK - GROUP CONSISTS
0472  E1                POP  H PUT PTR TO SIGN-OF-STEP IN HL, DE HAS PTR TO VAR VALUE (FROM STACK)   OF 17 BYTES)
0473  C8 <-'NEXT' or 'FOR' with var  RZ RETURN IF PROCESSING 'FOR' + ENTRY ON STACK MATCHES OR PROCESSING A 'NEXT' INSTRUCTION
           match     found on stack
0474  09                DAD  B   LOOK AT EARLIER ENTRY ON STACK FOR VAR PTR TO MATCH VAR PTR OF CURRENT
0475  C3 5B 04          JMP  X045B LOOP UNTIL NO MORE 'FOR' GROUPS ON STACK           CC 'FOR'
-------------------------------------------------------------------------------------------------
0478  CD 94 04 X0478    CALL X0494 RET HERE ONLY IF ENOUGH MEMORY FOR LINE IN BUFR TO BE  M   STORED IN STMT AREA
047B  C5       X047B    PUSH B } SWAP BC+HL              [OPEN N BYTES IN MEMORY      E
047C  E3                XTHL   } BC = new-nxt-byte       [FOR NEW STMT TO BE                AFTER B.C + H.L SWAPPED
047D  C1                POP  B } HL = former nxt byte    [INSERTED                    A
047E  E7      ->X047E   RST  4   TEST FOR PTR EQUALITY (RZ WHEN HL = DE)
047F  7E                MOV  A,M } MOVE 1 BYTE INTO HIGHER MEMORY
0480  02                STAX B   }
0481  C8                RZ       RETURN IF HL = DE (LAST BYTE HAS BEEN MOVED)         H
0482  0B                DCX  B } ADJUST PTRS TO MOVE THE NEXT BYTE
0483  2B                DCX  H }                                                      +
0484  C3 7E 04          JMP  X047E MOVE ANOTHER BYTE                                  C
-------------------------------------------------------------------------------------------------
0487  E5       X0487    PUSH H     SAVE PTR IN STACK
0488  2A E5 03          LHLD X03E5 FETCH PTR STORED IN 03E5 (ADDR OF BYTE AFTER LAST BYTE OF ARRAY, TABLE)
048B  06 00             MVI  B,H'00' } MULTIPLY VALUE IN C REG BY 2
048D  09                DAD  B       } AND ADD TO HL
048E  09                DAD  B       }
048F  CD 94 04          CALL X0494 TEST IF ENOUGH MEMORY IS AVAILABLE                 M
0492  E1                POP  H     RESTORE PTR, FETCH FROM STACK
0493  C9                RET                                                           I
-------------------------------------------------------------------------------------------------
0494  D5       X0494    PUSH D  save ptr for nxt byte in stmt area (bufr stmt will start at this addr)  SAVE DE IN STACK
0495  EB                XCHG    D,E = addr at end of stmt area after bufr inserted
0496  21 D2 FF          LXI  H,XFFD2 } TEST THAT NEW-NXT-BYTE IS LESS THAN        IR
0499  39                DAD  SP      } (SP)-46  (STACK NEEDS 46 LOC'S NOT AVAILABLE  9
049A  E7                RST  4       }           TO STMT AREA)
```



```
                                                                                         1020
0490  EB                 XCHG   } RESTORE ADDR of byte to put bufr into stmt area   C   1021  RESTORE DE FROM STACK
049C  01                 POP  D }                                                   P   1022
0490  00                 RNC    RNC IF HL ≥ DE (RNC IF SP-46₁₀ ≥ new end of stmt area)
                                                                   E  ERROR MESSAGE      1023  ERROR MESSAGE PROCESSING
049E  1E 07     X049E    MVI  E,H'07' } ERROR MESSAGE "OUT OF MEMORY"  Ø1 NEXT WITHOUT FOR C7  1024
04AC  C3 B7 04           JMP  X04B7   }                                 Ø7 OUT OF MEMORY   *O   1025
04A3  2A CF 03  X04A3    LHLD X03CF } SAVE ADDR OF LINE #   LOAD E REGISTER   ØA REDIMENSIONED ARRAY*H  1026
04A6  22 D7 03           SHLD X0307 } WHERE ERROR OCCURS    WITH A NUMBER     Ø8 DIVISION BY ZERO   1027
04A9  1E 02     X04A9    MVI  E,H'02'                                  12 UNDEFINED USER FUNCTION  1028
04AB  01 1E 0B           LXI  B,X0B1E MVI E,H'Ø8'                      Ø2 SYNTAX ERROR     1029
04AE  01 1E 01           LXI  B,X011E MVI E,H'Ø1'                                          1030
04B1  01 1E 0A           LXI  B,X0A1E MVI E,H'ØA'                                          1031
04B4  01 1E 12           LXI  B,X121E MVI E,H'12'                                          1031
04B7  CD CC 05  X04B7    CALL X05CC RESET PTRS (PART OF NEW)                          HL   1032
04BA  AF                 XRA  A     } CLEAR PRINT SUPPRESSION FLAG                    /    1033
04BB  32 A2 03           STA  X03A2 }                                                 2"   1034
04BE  CD 98 0A           CALL X0A98 OUTPUT CR LF + NULLS                              M    1035
04C1  21 19 02           LXI  H,X0219 HL POINT TO BYTE BEFORE START OF ERROR MESSAGE ASCII TABLE  ! (=NULL)  1036
04C4  7B                 MOV  A,E   } SAVE ERROR TOKEN - IF Ø2 - EXECUTE EDIT ROUTINE                1037
04C5  32 E9 03           STA  X03E9 }                 IF "SYNTAX ERROR" SEE Ø4E8   2   1038
04C8  CD 8E 09  X04C8    CALL X098E SCAN ERROR MESSAGE STRING TIL NULL FOUND          M    1039
04CB  1D                 DCR  E DECREMENT ERROR MESSAGE COUNTER                            1040
04CC  23                 INX  H ADVANCE PTR TO POINT TO 1ST BYTE OF NXT ERROR MESSAGE  #   1041
04CD  C2 C8 04           JNZ  X04C8 JMP IF NULL-COUNT NOT ZERO YET                    BH   1042
04D0  CD 41 12  X04D0    CALL X1241 PRINT THE ASCII STRING                            MA   1043
04D3  2A D7 03           LHLD X0307 FETCH LINE #                                      *W   1044
04D6  7C                 MOV  A,H } IF HL = FFFF ZERO FLAG SET                             1045
04D7  A5                 ANA  L   } (CURRENT ADDR LOC USED AS FLAG FOR CONDITIONAL CALL - see Ø4FD)  %  1046
04D8  3C                 INR  A   }                                                   <    1047
04D9  C4 6B 21           CNZ  X216B PRINT ASCII STRING " IN XXXXX" XXXXX = LINE #     D I  1048
04DC  3E C1              MVI  A,H'C1'                                                 >A   1049
04DE  AF        X04DE    XRA  A     CLEAR ACC                                         /    1050  BASIC COMMAND MODE
04DF  32 A2 03           STA  X03A2 CLEAR Ø3A2 TO ØØ                                  2"   1051  PROCESSING
04E2  21 48 04           LXI  H,X0448 LOAD HL WITH LOC OF CR LF "OK" CRLF             !H   1052
04E5  CD 90 27           CALL X2790 INITIALIZATION SUBR (AFTER INITIALIZATION THIS IS CHANGED TO  M 'TO  1053  CALL X1241)
04E8  3A E9 03           LDA  X03E9 } EXECUTE EDIT ROUTINE IF LAST ERROR MESSAGE      :    1054
04EB  D6 02              SUI  H'02' } WAS "SYNTAX ERROR"                              V    1055
04ED  CC FE 16           CZ   X16FE                                                   L    1056
04F0  21 FF FF  X04F0    LXI  H,XFFFF } SET Ø3D7 + Ø3D8 TO FFFF } USED TO PREVENT INPUT   !    1057
04F3  22 D7 03  X04F3    SHLD X03D7   }                          } AS A DIRECT COMMAND   "H   1058
04F6  CD 94 06           CALL X0694 FILL INPUT BUFR WITH LINE FROM INPUT CONSOLE      M    1059
04F9  D7                 RST  2 MOVE H,L TO POINT TO FIRST NON-SPACE CHAR IN INPUT BUFR  W  1060
04FA  3C                 INR  A } TEST THIS FIRST NON-SPACE CHAR FOR BEING A NULL     <    1061
04FB  3D                 DCR  A }                                                     =    1062
04FC  CA F0 04           JZ   X04F0 DO NOT PROCESS LINE IN BUFFER IF ALL SPACES (1ST NON-SPACE  1063  CHAR IS A NULL)
04FF  F5                 PUSH PSW SAVE CARRY FLAG  C = #   NC = CHAR (NC IF IMMEDIATE INSTR  1064  EXECUTE, NO LINE #)
0500  CC F3 08           CALL X08F3 REMOVE + CONVERT LINE #, HL PT TO NEXT NON-SPACE CHAR  H  1065  IN BUFR
0503  D5    EDIT→X0503   PUSH D   SAVE LINE # IN STACK                                U    1066
0504  CD EF 05           CALL X05EF CONDENSE LINE IN BUFFER, HL RESET TO PT TO BUFR   M    1067  START -1
0507  47                 MOV  B,A   ZERO B REG, C REG IS CHAR COUNT                   G    1068
0508  D1                 POP  D     RESTORE LINE # TO D,E                             Q    1069
0509  F1                 POP  PSW   RESTORE CARRY FLAG                                     1070
050A  D2 33 07           JNC  X0733 EXECUTE IMMEDIATE INSTRUCTION                     R3   1071
050D  D5                 PUSH D BUFR LINE # TO STACK                                  U    1072
050E  C5                 PUSH B   SAVE CHAR COUNT OF CONDENSED BUFR                   E    1073
050F  D7                 RST  2 } FIND + TEST 1ST BYTE IN CONDENSED BUFR              H    1074
0510  B7                 ORA  A } (TEST FOR NULL)                                     7    1075
0511  F5                 PUSH PSW SAVE FLAGS                                               1076  install line from condensed bufr
0512  CD 7B 05           CALL X057B Test stmt Area to determine where in stmt area to M    1077
0515  C5                 PUSH B PUT ADDR OF LO BYTE OF CHAIN ADDR IN STACK (PTR TO WHERE LINE  E  1078  FROM BUFR IS TO BE STORED IN STMT AREA
0516  DC 39 15           CC   X1539 CARRY SET IF BUFR LINE # = STMT AREA LINE #      \9   1079  (FORMER LINE MUST BE REMOVED BEFORE
                 A B C D        REMOVE LINE FROM STMT AREA, ADDR OF 1ST BYTE FOR NEW LINE IN STACK       INSERTING NEW LINE)
```

```
                       A B C D  PTR TO ADDR TO PUT BUFR IN      STMT AREA
0519  D1                 POP  D   PUT ADDR OF CHAIN ADDR IN D,E (EITHER END OF STMT AREA OR   WHERE IN MIDDLE OF STMT AREA  CONDENSED
051A  F1                 POP  PSW RESTORE ZERO FLAG                                             LINE MUST BE WEDGED IN
051B  CA 40 05           JZ   X0540 null (line # 0 was typed) reset ptrs + flags as part of JZ  "NEW"
051E  2A E1 03           LHLD X03E1  H,L = addr of nxt-byte in stmt area
0521  E3                 XTHL put addr. of nxt-byte in stack, B is Ø   C is byte count of line in   condensed bufr  } HL=char count
                              swap HL & BC                                                        } BC=addr of nxt-byte
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 new nxt-available byte in stack
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 ADJUSTMENT)
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  6B                 MOV  L,E }    HL TO SCAN
0547  7E                 MOV  A,M }
0548  23                 INX  H   } TEST BOTH BYTES OF CHAIN ADDR IN STMT AREA
0549  B6                 ORA  M   }
054A  CA F0 04           JZ   X04F0 IF NULL, ENTIRE STMT AREA PROCESSED. FETCH ANOTHER LINE OR COMMAND FROM TTY
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 6E 05  RSRVD WRD NO ARGS <- JZ  X056E JMP IF RESERVED WRD FOLLOWED BY A ':' or a null
0564  D1                 POP  D  REMOVE ADDR FROM STACK
0565  CC F3 08           CALL X08F3 ASSEMBLE NXT CHAR(S) INTO A PACKED 2-BYTE VALUE - IGNORE SPACES
0568  D5                 PUSH D  PUT LINE # ON STACK (1ST ARG)
0569  CA 77 05  RSRVD WRD 1 ARG <- JZ  X0577 IF NON-SPACE CHAR AFTER LAST DIGIT IS A NULL or colon, JMP (only 1 numerical argument to instruction)
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 2 INSTRUCTIONS
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
```



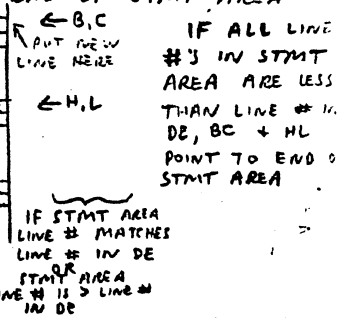
45

```
0577  E3          X0577  XCHG        } SWAP RET ADDR & LINE #
0578  D1                 POP  D      } ON STK
0579  E3                 XTHL
057A  E5                 PUSH H      } LINE # IN DE   HL DON'T CARE

057B  2A 0F 03     X057B  LHLD X030F  H,L ← FIRST ADDR IN STMT AREA     D&E = LINE # of line in condensed buffer
057E  44           X057E  MOV  B,H    } SAVE ADDR IN B,C (TEMPORARY STORAGE)
057F  4D                  MOV  C,L
0580  7E                  MOV  A,M    LOAD LO BYTE OF CHAIN ADDR
0581  23                  INX  H      ADVANCE STMT AREA PTR
0582  B6                  ORA  M      OR HI BYTE OF CHAIN ADDR INTO ACC
0583  2B                  DCX  H      MOVE STMT AREA PTR BACK 1
0584  C8                  RZ          IF CHAIN ADDR IS ØØ IN STMT AREA, RETURN
0585  23                  INX  H      } ADVANCE STMT AREA PTR TO POINT TO LO BYTE OF LINE #
0586  23                  INX  H      }
0587  7E                  MOV  A,M    } PUT LINE # (16 BIT VALUE) IN H+L
0588  23                  INX  H      } (LINE # of line in stmt area)
0589  66                  MOV  H,M    }
058A  6F                  MOV  L,A    }
058B  E7                  RST  4      RZ HL=DE   RNC HL<DE   RNZ HL>DE
058C  60                  MOV  H,B    } ADDR IN STMT AREA (PTS TO LO BYTE OF CHAIN ADDR) is 1st byte
058D  69                  MOV  L,C    } BC=HL
058E  7E                  MOV  A,M    } PUT CHAIN ADDR IN H,L (ADDR OF 1ST BYTE OF NXT LINE IN STMT AREA)
058F  23                  INX  H      }
0590  66                  MOV  H,M    }
0591  6F                  MOV  L,A    }
0592  3F                  CMC
0593  C8                  RZ          RETURN IF LINE # OF LINE IN CONDENSED BUFR = LINE # POINTED
                                      CARRY SET IF RZ & HL=DE
0594  3F                  CMC
0595  D0                  RNC         RETURN IF LINE # OF LINE IN CONDENSED BUFR < LINE # POINTED
0596  C3 7E 05            JMP  X057E  TEST NEXT LINE IN STMT AREA

NEW→0599  C0              RNZ         IF "NEW" TOKEN FOLLOWED BY ANY CHAR NOT A NULL, COLON, SPACE - WRONG
059A  2A CF 03     X059A  LHLD X03CF  FETCH ADDR OF START OF STMT AREA INTO HL
059D  CD 3F 08            CALL X083F  SETS LOC ØØ4C TO ØØ - TURN TRACE FLAG OFF
05A0  77                  MOV  M,A    } PUT A DOUBLE NULL IN FIRST TWO LOC'S IN STMT AREA
05A1  23                  INX  H      } ADVANCE PTR BY 2
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 0F 03     X05A7  LHLD X030F  } SET ADDR'S Ø3D3 & Ø3D4 TO POINT TO ADDR
05AA  2B                  DCX  H      } ONE LESS THAN START OF STMT AREA (START PROG AT 1ST STMT)
05AB  22 03 03     X05AB  SHLD X0303  }
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  3E 04        X05B3  MVI  M,H'04' } PUT Ø4 IN LOC Ø3EA - Ø4D3 (26 LOC TOTAL)
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  }
05BA  2A A6 03            LHLD X03A6  } ELIMINATE ALL STRINGS STORED IN STRING SPACE
05BD  22 C0 03            SHLD X03C0  }
05C0  CD E1 07            CALL X07E1  RESET PTR FOR NXT (OR CURRENT) DATA STMT TO START OF PGM
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 OF RETURN ADDR FROM STACK
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
```

BEFORE STK: RET ADDR, LINE #
AFTER STK: LINE #

FIND STMT IN STMT AREA

TEST FOR 2 NULLS IN A ROW INDICATES END OF STMT AREA



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

TO BY B,C (START H,L AT ADDR IN Ø3DF + Ø3EØ)

TO BY B,C

SYNTAX

NEW

ELIMINATE PROGRAM BY RESETTING FLAGS FOR STMT AREA

ENTER HERE - RUN WITH NO ARGUMENT

SAME AS RESTORE

```
0509  L.                 MOV  L,A  CLEAR L
050A  22 00 03           SHLD X0308 CLEAR PTR TO 0000                                            "(
050D  32 01 03           STA  X0301 CLEAR LOC 0301 TO 00 - CLEAR FLAG LOC                         2Q
05E0  E5                 PUSH H  PUT 0000 ON STACK
05E1  C5                 PUSH B  PUT RETURN ADDR BACK ON STACK                                    E
05E2  2A 03 03   X05E2   LHLD X0303  RESTORE H,L TO PT TO ONE LESS THAN START OF STMT AREA*S
05E5  C9                 RET                                                                      I
-------------------------------------------------------------------------------------------------
05E6  3E 3F      X05E6   MVI  A,A'?'  } OUTPUT "?"                                                >?   OUTPUT "?␣"
05E8  DF                 RST  3       }                                                           _
05E9  3E 20              MVI  A,A' '  } OUTPUT "␣"                                                >    INPUT FROM TTY
05EB  DF                 RST  3       }                                                           _
05EC  C3 34 06           JMP  X0634  JMP TO TTY INPUT ROUTINE                                     C
-------------------------------------------------------------------------------------------------
05EF  AF         X05EF   XRA  A      } CLEAR TEMP STORAGE BYTE                                    /
05F0  32 A5 03           STA  X03A5  }                                                            2%
05F3  0E 05              MVI  C,H'05'  START CHAR COUNT AT 5 (2 FOR CHN ADDR, 2 FOR PACKED LINE #, 1 FOR NULL AT END)
05F5  11 5A 03           LXI  D,X035A START PTR AT FIRST CHAR IN INPUT BUFR                       Z
05F8  7E         X05F8   MOV  A,M  FETCH CHAR FROM BUFR
05F9  FE 20              CPI  A' '   } IF CHAR IS A SPACE, PUT IN CONDENSED BUFR
05FB  CA 4D 06           JZ   X064D  }                                                            JM
05FE  47                 MOV  B,A  SAVE CHAR IN B REG, IF CHAR IS NULL CLEAR B REG                G
05FF  FE 22              CPI  A'"'   } IF CHAR IS ", PUT CHAR IN CONDENSED BUFR TIL NEXT "        "
0601  CA 6D 06           JZ   X066D  }                                                            J
0604  B7                 ORA  A      } IF CHAR IS NULL, END OF PROCESSING TO CONDENSE BUFR        7
0605  CA 81 06           JZ   X0681  }                                                            J
0608  3A A5 03           LDA  X03A5  } RELOAD ACC FROM TEMP STORAGE  {03A5 ≠ 0 IF PROCESSING      %
060B  B7                 ORA  A      }                                A DATA STMT                 7
060C  47                 MOV  B,A  SAVE CHAR IN B                                                 G
060D  7E                 MOV  A,M  FETCH SAME CHAR AGAIN FROM BUFR
060E  C2 4D 06           JNZ  X064D  JMP IF (03A5) ≠ 0 KEEP PUTTING CHAR IN CONDENSED BUFR TIL NULL OR COLON FOUND   BM
0611  FE 3F              CPI  A'?'    } IF CHAR IS '?' [PRINT] PUT CODE BYTE IN                   ?
0613  3E A0              MVI  A,H'A0' }    CONDENSED BUFR                                         >
0615  CA 4D 06           JZ   X064D   }                                                           JM
0618  7E                 MOV  A,M  FETCH CHAR FROM BUFR AGAIN
0619  FE 30              CPI  A'0'   } TEST + JMP IF CHAR IS LESS THAN A '0'                      0
061B  DA 23 06           JC   X0623  }                                                            Z0
061E  FE 3C      X061E   CPI  A'<'   } TEST CHAR, IF DIGIT 0-9, or ':' ';' PUT BYTE IN            <
0620  DA 4D 06           JC   X064D  }    CONDENSED BUFR                                          ZM
-------------------------------------------------------------------------------------------------
0623  D5       ->X0623   PUSH D SAVE PTR TO WRD IN BUFR, TRY TO DETERMINE IF WRD IS A RESERVED WRD  U
0624  11 79 00           LXI  D,X0079 USE DE TO PT TO RESERVED WRD LIST, HL PTS TO WRD IN BUFR
0627  E5                 PUSH H SAVE PTR TO CONDENSED BUFR, SKIP RST 2 TO GET INTO LOOP
0628  3E 07 DUMMY BYTE -> RST 2   MVI  A,H'D7'                                                    >H
062A  13                 INX  D ADVANCE RESERVED WRD PTR TO NEXT CHAR IN WRD
062B  1A         X062B   LDAX D FETCH RESERVED WRD CHAR
062C  E6 7F              ANI  H'7F'  } TEST CHAR FROM RESERVED WRD LIST, IF END OF LIST JMP
062E  CA 3F 06           JZ   X063F  }   (no more reserved wrds)                                  J?
0631  BE                 CMP  M COMPARE RESERVED WRD CHAR WITH CHAR OF WRD IN BUFR                >
0632  C2 74 06           JNZ  X0674 NO MATCH, MOVE PTRS TO LOOK AT NXT RESERVED WRD               B
0635  1A                 LDAX D   } FETCH RESERVED WRD CHAR
0636  B7                 ORA  A   } TEST D7 BIT                                                   7
0637  F2 29 06           JP   X0629 } IF D7 BIT IS 0, NOT THE LAST CHAR OF RESERVED, TRY TO MATCH NXT CHAR   )
063A  F1                 POP  PSW RESERVED WRD TOTAL MATCH, POP PSW REMOVES PTR (ON STACK) TO START OF WRD IN BUFR JUST MATCHED
063B  78                 MOV  A,B PUT RESERVED WRD FAIL-TO-MATCH CTR IN ACC
063C  F6 80              ORI  H'80' ADD 80 TO FORM CODE BYTE FOR RESERVED WRD (ORI SETS FLAG 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
0641  D1                 POP  D RESTORE DE TO POINT TO CONDENSED BUFR                             Q
-------------------------------------------------------------------------------------------------
0642  EB                 XCHG  EXCHANGE REGISTERS
0643  FE 90              CPI  H'90'  } IF RESERVED WRD IS 'ELSE', PUT ':' IN CONDENSED BUFR THEN PUT ELSE TOKEN = 90H
0645  36 3A              MVI  M,A':' }                                                            6:
0647  C2 4C 06           JNZ  X064C                                                               BL
064A  0C                 INR  C  ADVANCE CHAR COUNT
```

```
0649 23          INX  H ADVANCE PTR TO CONDENSED BUFR                                   #
064C EB   X064C  XCHG  EXCHANGE REGISTERS AGAIN; HL PT TO BUFR, DE PT TO CONDENSED        BUFR
064D 23   X064D  INX  H ADVANCE PTR TO LOOK AT NXT CHAR IN BUFR                          #
064E 12          STAX D STORE CHAR IN BUFR
064F 13          INX  D ADVANCE CONDENSED BUFR PTR
0650 0C          INR  C INCREMENT CHAR COUNT
0651 D6 3A       SUI  A':' } IF CHAR IS A COLON - STORE IN CONDENSED BUFR-FETCH NXT CHAR V:
0653 CA 5B 06    JZ   X065B }                                                             J(
0656 FE 49       CPI  A'I' }IF TOKEN(KEYWORD CODE) IS 83 ='DATA', STORE NON-ZERO # IN 03A5 I (NOTE: CPI DOES NOT ZERO ACC)
0658 C2 5E 06    JNZ  X065E                                                               D^
065B 32 A5 03 X065B STA X03A5 SET 03A5 TO NON-ZERO IF DATA STMT, SET BACK TO ZERO AT NXT ':' FOUND   2%
065E D6 54 X065E SUI  A'T' }IF TOKEN IS 8E = 'REM'                                        VT
0660 C2 F8 05    JNZ  X05F8 IF NOT REM - WORK ON NXT CHAR IN LINE BUFR                    B
0663 47          MOV  B,A PUT CHAR IN B (REM WILL GET STUCK IN NXT LOOP TIL NULL)         G
0664 7E   X0664  MOV  A,M } IF CHAR IS A NULL, END OF PROCESSING TO CONDENSE BUFR
0665 B7          ORA  A                                                                   7
0666 CA 81 06    JZ   X0681 }                                                             J
0669 B8          CMP  B  TEST CHAR FOR 2ND "                                              8
066A CA 4D 06    JZ   X064D                                                               JM
066D 23   X066D  INX  H  ADVANCE PTR TO BUFR                                              #
066E 12          STAX D  STORE CHAR IN CONDENSED BUFR
066F 0C          INR  C  INCR CHAR COUNT
0670 13          INX  D  ADVANCE CONDENSED BUFR PTR
0671 C3 64 06    JMP  X0664 STAY IN THIS LOOP TIL 2ND " FOUND, OR IF REM, MOVE REST OF CBUFR INTO CONDENSED BUFR TIL NULL FOUND
0674 E1   X0674  POP  H } RESET PTR TO POINT TO FIRST CHAR OF WRD IN BUFR FOR NXT ATTEMPTED RESERVED WRD MATCH
0675 E5          PUSH H }
0676 04          INR  B ADVANCE RESERVED WRD FAIL-TO-MATCH CTR
0677 EB          XCHG EXCHANGE REGISTERS SO HL POINT TO RESERVED WRD LIST
0678 B6   X0678  ORA  M  FETCH CHAR OF RESERVED WRD                                       6
0679 23          INX  H ADVANCE PTR TIL POINTING TO FIRST BYTE OF NXT RESERVED WRD LIST   #
067A F2 78 06    JP   X0678 LAST CHAR OF RESERVED HAS D7 SET (JMP MINUS)
067D EB          XCHG EXCHANGE REGISTERS AGAIN
067E C3 20 06    JMP  X0620 TEST NXT RESERVED FOR A MATCH                                 C+
0681 21 59 03 X0681 LXI H,X0359 RESET PTR TO POINT AT BYTE BEFORE START OF BUFR           !Y
0684 12          STAX D }
0685 13          INX  D }  STORE 3 NULLS AT END OF CONDENSED BUFR
0686 12          STAX D }
0687 13          INX  D }
0688 12          STAX D }
0689 C9          RET     END OF STMT PROCESSING                                            I
----------------------------------------------------------------------------------------------
068A 05 DEL CHAR X068A DCR B } DELETE 1 CHAR (SUBTRACT 1 FROM CHAR COUNT & LINE BUFR PTR)   GET CHAR FROM
068B 2B          DCX  H }                                                          +         INPUT CONSOLE
068C DF          RST  3  ECHO CHAR (BACKARROW)                                     _         & PROCESS IT
068D C2 99 06    JNZ  X0699                                                        B
0690 DF END OF LINE X0690 RST 3  ECHO CHAR                                         _
0691 CD 9B 0A    CALL X0A9B PRINT CRLF + NULLS                                     M
----------------------------------------------------------------------------------------------
0694 21 5A 03 X0694 LXI H,X035A SET HL TO POINT TO FIRST BYTE OF LINE BUFR         !Z
0697 06 01       MVI  B,H'01' INITIALIZE CHAR CTR TO 1
0699 CD EC 06 X0699 CALL X06EC GET A CHAR FROM KEYBOARD                            M
069C FE 07       CPI  H'07' } TEST FOR CHAR = CNTL G (BELL)
069E CA DA 06    JZ   X06DA }                                                      J:
06A1 FE 0D       CPI  H'0D' }  CHAR = CR
06A3 CA 93 0A    JZ   X0A93 }                                                      J
06A6 FE 20       CPI  A' ' } IF ANY OTHER CNTL CHAR, IGNORE IT
06A8 DA 99 06    JC   X0699 }                                                      Z
06AB FE 7D       CPI  H'7D' } IF ALT MODE CHAR, IGNORE IT
06AD D2 99 06    JNC  X0699 }                                                      R
06B0 FE 40       CPI  A'@' } TERMINATES LINE INPUT & CANCELS LINE IN BUFR          @
06B2 CA 90 06    JZ   X0690 }                                                      J
06B5 FE 5F       CPI  A'_' } IF CHAR IS A BACKARROW, DELETE CHAR                   _
```

```
06B7  CA 8A 06          JZ    X068A  } IF CHAR IS A BACKARROW, DELETE IT
06BA  4F          X06BA 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 03A2), 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 0027
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 03A2 = 1 NO PRINTING
06EB  C9                RET
-------------------------------------------------------------------------------
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
-------------------------------------------------------------------------------
0702  3E 64            MVI   A,H'64'} STORE 64 AT LOC 0301
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 07 03          LHLD  X0307} 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 09

TERMINAL WIDTH

PRINT AN ASCII CHAR

CHAR TO PRINT IS ON STACK

RST 3 falls into this routine

INPUT A CHAR FROM INPUT CONSOLE

RETURN WITH CHAR IN ACC

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

bh

```
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
074F  CD 51 18        CALL  X1851 PUT FLOATING-PT VALUE FOR 2ND EXPRESSION IN BCDE, B=EXPONENT
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 18        CALL  X1851 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, 0, 1 => NEG, 0, 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 0303)
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     X0770 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,
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 93 07        JZ    X0793 }
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 CNTL-C CHECK
1429
1430 IF CHAR NOT A CNTL-C
1431
1432
1433
1434
1435
1436
1437
1438
1439
```

```
I
R
MR
L
H
STACK
L
U
M/
C
HL
MQ BYTE
E
U
Q
Z
-
>
B
MS
MQ
E
U
O
G
E
*S
E
"S
:
JJ
7
B)
#
#
```

STACK

INTEGER — SNG-PREC
'FOR' TERMINATOR ADDR
'FOR' LINE #
2ND EXPRESSION — 2ND EXPRESSION
3RD EXPRESSION
4 bytes — 3RD EXPRESSION
(TYPE-3) = FF — (TYPE-3) = 01
SIGN OF STEP — SIGN OF STEP
PTR TO VAR VALUE REGION
'FOR' TOKEN

TOTAL = 17 BYTES FOR EACH ACTIVE 'FOR'

```
                                }TEST 2 BYTES AFTER NULL
0794  B6              ORA   H  }FOR DOUBLE NULL                                      6
0795  23              INX   H ADVANCE STMT PTR TO POINT TO FIRST BYTE OF LINE # OF NXT STMT
0796  CA FC 07        JZ    X07FC JMP IF END OF STMT AREA (DO A 'STOP' - TERMINATE PROGRAM)
0799  5E              MOV   E,M }FETCH LINE # OF NXT STMT INTO DE                     ^
079A  23              INX   H   }                                                    #
079B  56              MOV   D,M }                                                    V
079C  EB              XCHG      }STORE LINE # AT LOC'S Ø307 + Ø308, PTR TO STMT IN DE
079D  22 07 03        SHLD  X0307}                                                   "H
07A0  3A 0C 04        LDA   X040C FETCH TRACE FLAG                                   :
07A3  B7              ORA   A    }IF TRACE FLAG IS Ø, EXECUTE NEXT INSTRUCTION        7
07A4  CA B2 07        JZ    X07B2} TRACE IS OFF IF FLAG = Ø                          J2
07A7  D5              PUSH  D SAVE PTR TO STMT ON STACK                              U
07A8  3E 5B           MVI   A,A'['} PRINT '[' CHAR                                   >(
07AA  DF              RST   3     }                                                  _
07AB  CD 73 21        CALL  X2173 CONVERT LINE # (IN HL) TO DECIMAL ASCII + PRINT     M !
07AE  3E 5D           MVI   A,A']'} PRINT ']' CHAR                                   >)
07B0  DF              RST   3     }                                                  _
07B1  D1              POP   D FETCH PTR TO STMT FROM STACK                           Q
07B2  EB     X07B2    XCHG  PUT PTR TO STMT BACK IN HL
-------------------------------------------------------------------------------------------
07B3  D7     X07B3    RST   2 SCAN CONDENSED BUFR, RZ IF NULL or colon               W    COMPUTE SUBROUTINE ADDR
07B4  11 7D 07        LXI   D,X077D} CNTL C CHECK                                         OF INSTRUCTION + EXECUTE
07B7  D5              PUSH  D                                                        U    INSTRUCTION BY BRANCHING
07B8  C8     X07B8    RZ    Return if colon pointed to by H,L                        H    TO THAT SUBROUTINE
07B9  D6 80  X07B9    SUI   H'80' SUBTRACT CODE WRD OFFSET                           V
07BB  DA AB 09        JC    X09AB JMP IF NOT A RESERVED WRD, JMP TO LET ROUTINE      Z+
07BE  FE 26           CPI   A'&' CODE WRD A6 [TAB(] + LARGER NOT ALLOWED TO BE FIRST  &   ON A LINE
07C0  D2 A9 04        JNC   X04A9 IF A6 or larger, JMP + PRINT SYNTAX ERROR          R)
07C3  07              RLC     MULTIPLY BY 2 (RESERVED WRD ADDR TABLE IS 2 BYTES FOR       EACH RESERVED WRD)
07C4  4F              MOV   C,A                                                      O
07C5  06 00           MVI   B,H'00' } BC = [(CODE WRD) - 80H] * 2
07C7  EB              XCHG  SAVE PTR TO INSTRUCTION STRING BEING EXECUTED, IGNORE DE
07C8  21 A5 01        LXI   H,X01A5 HL = ADDR OF RESERVED WRD ADDR TABLE             !%
07CB  09              DAD   B CALCULATE ADDR TO FETCH ADDR FROM TABLE
07CC  4E              MOV   C,M }                                                    N
07CD  23              INX   H   }FETCH ADDR FROM TABLE INTO B,C                      #      IF ENTER FROM HERE
07CE  46              MOV   B,M }                                                    F    SET CARRY or NOT CARRY
07CF  C5              PUSH  B PUT ADDR OF SUBROUTINE ON STACK                        E    DEPENDENT ON FIRST NON-SPACE
07D0  EB              XCHG  HL POINTS TO INSTRUCTION STRING                                 BYTE AFTER RESERVED WRD IN ST
07D1  23     X07D1    INX   H  ADVANCE STRING PTR                                    #
07D2  7E              MOV   A,M  FETCH NEXT BYTE FROM INST STRING
07D3  FE 1A           CPI   A'!'                                                     :
07D5  D0              RNC   RNC IF '!' or larger                                     P
07D6  FE 20  X07D6    CPI   A' ' }                                                        
07D8  CA D1 07        JZ    X07D1}   SKIP OVER ANY SPACES AFTER RESERVED WRD CODE    JQ   BYTE
07DB  FE 30           CPI   A'0'                                                     0
07DD  3F              CMC                                                            ?
07DE  3C              INR   A   JMP TO SUBROUTINE FOUND IN ADDR TABLE                <
07DF  3D              DCR   A  {RNC IF NON-SPACE CHAR AFTER CODE BYTE IS NOT A       =    DIGIT
07E0  C9              RET      {RC  IF NON-SPACE CHAR AFTER CODE BYTE IS A DIGIT     I    RZ IF CHAR IS A NULL
-------------------------------------------------------------------------------------------
07E1  EB     X07E1    XCHG        SAVE HL IN DE                                           RESTORE
07E2  2A 0F 03        LHLD  X030F LOAD START ADDR OF STMT AREA                       *_
07E5  2B              DCX   H     DECREMENT BY 1                                     +
07E6  22 E7 03 X07E6  SHLD  X03E7 STORE ADDR AT Ø3E7 + Ø3E8                          "
07E9  EB              XCHG        RESTORE HL TO PREVIOUS VALUE
07EA  C9              RET                                                            I
-------------------------------------------------------------------------------------------
07EB  DB 00  X07EB    IN    H'00' INPUT TTY STATUS WRD                               {
07ED  E6 01           ANI   H'01' TEST FOR CHAR HAVING BEEN TYPED
07EF  C0              RNZ   RETURN IF CHAR NOT TYPED                                 @
07F0  CD EC 06 X07F0  CALL  X06EC INPUT A CHAR FROM INPUT CONSOLE, CHAR IN ACC        M
07F3  FE 03           CPI   H'03' IS CHAR CNTL-C ?
```

```
07F5  C0                RNZ  IF CHAR IS NOT A CNTL-C, RETURN
07F6  F6 C0     DUMMY   ORI  #'C0'
07F8  22 03 03          SHLD X0303  SAVE PTR TO CURRENT INSTRUCTION (USED BY CONT)
07FB  C1                POP  B  REMOVE 1 ADDR FROM STACK (0770 IF STOP or END, 0784 IF CNTL-C TYPED)
07FC  F5        X07FC   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        X0800   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        X080F   XRA  A  } CLEAR PRINT SUPPRESSION FLAG
0810  32 12 03          STA  X0312 }
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 LINE #
081A  C3 0E 04          JMP  X040E NO MESSAGE IF 'END' INSTR, RETURN TO BASIC COMMAND MODE

081D  1E 11             MVI  E,#'11' ERROR MESSAGE # IN E
081F  2A 0B 03          LHLD X030B } TEST FLAG 030B + 030C FOR 0000
0822  7C                MOV  A,H  }
0823  B5                ORA  L    }
0824  CA 37 04          JZ   X0437 IF 030B IS 0000, 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 + RETURN
0834  3C                INR  A ADD 1 TO NULL COUNT
0835  FE 48             CPI  #'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,#'AF' } TRACE ON  { TRACE OFF
0840  32 0C 04          STA  X040C   } SET FLAG TO AF  { SET FLAG TO 00
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
0849  21 04 04          LXI  H,X0404 LOAD HL WITH ADDR OF 1ST BYTE OF AN 8 BYTE HOLDING AREA
084C  CD 66 18          CALL X1866 MOVE N BYTES FROM 1ST VARIABLE VALUE REGION TO HOLDING AREA
084F  2A E3 03          LHLD X03E3 } PUT PTR TO INSTRUCTION IN HL
0852  E3                XTHL       } PUT ADDR OF 1ST BYTE OF ARRAY TABLE ON STACK
0853  F7                RST  6  TEST VALUE TYPE OF 1ST ARGUMENT (ACC = TYPE-3)
0854  F5                PUSH PSW SAVE ACC ON STACK
0855  CF                RST  1  } SYNTAX CHECK FOR COMMA BETWEEN 1ST + 2ND ARGUMENTS
0856  2C                DB   ','  }
0857  CD E0 0E          CALL X0EE0 FIND 2ND VARIABLE OR ARRAY ELEMENT IN APPROPRIATE TABLE
085A  C1                POP  B FETCH 1ST ARG (TYPE-3) FROM STACK INTO B REG
085B  F7                RST  6 TEST VALUE TYPE OF 2ND ARGUMENT
085C  B8                CMP  B  } TEST IF BOTH ARGS ARE THE SAME TYPE
085D  C2 9A 1C          JNZ  X1C9A } IF NOT THE SAME TYPE, JMP, PRINT ERROR MESSAGE "TYPE MISMATCH"
0860  E3                XTHL PUT ADDR OF START OF ARRAY TABLE IN HL, PUT PTR TO INSTRUCTION ON STACK
0861  EB                XCHG } SAVE PTR TO 2ND ARG VARIABLE VALUE REGION ON STACK
0862  E5                PUSH H }
0863  2A E3 03          LHLD X03E3 LOAD HL WITH ADDR OF 1ST BYTE OF ARRAY TABLE AFTER 2ND VARIABLE
                        RST  4 TEST (HL) AGAINST (DE)
```

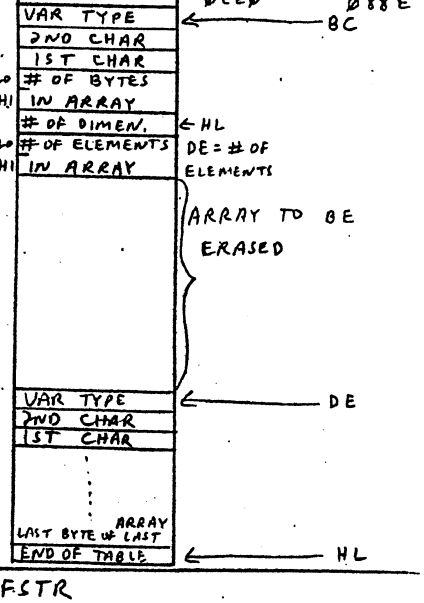
```
1500  STOP
1501  END
1502
1503
1504 RESET
1505
1506
1507
1508
1509 (0309 & 030B)
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520  CONT
1521
1522
1523 "CAN'T CONTINUE"
1524
1525
1526
1527
1528
1529                                  NULL
1530 + RETURN
1531
1532
1533
1534
1535
1536 (083E TR ON   083F TR OFF
1537
1538
1539  SWAP
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
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
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 18          CALL  X1866 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 18          CALL  X1866 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 01 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=00 SEE 1013)
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 CHAIR
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   (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
```

BA 1560 "ILLEGAL FUNCTION CALL"

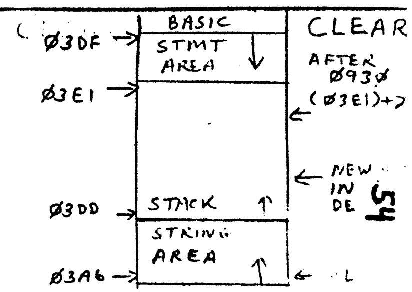
ERASE

ARRAY FOUND IN ARRAY TABL



DEFSTR
DEFINT
DEFSNG
DEFDBL

```
0BCD  D7                 RST  2  SCAN FOR NEXT CHAR                                                  W   1620
0BCE  78'       X0BCE    MOV  A,B PUT 2ND ARG OFFSET IN ACC                                              1621
0BCF  91                 SUB  C  SUB 1ST ARG OFFSET                                                      1622
0BD0  D8                 RC   SYNTAX ERROR IF 2ND IS SMALLER THAN 1ST  ie  B-A                         X   1623
0BD1  3C                 INR  A  ADJUST ACC TO # OF VARIABLE-LETTERS TO RE-DEFINE                      <   1624
0BD2  E3                 XTHL REMOVE SYNTAX ERROR ADDRESS FROM STACK, PUT PTR TO STMT ON STACK             1625
0BD3  21 EA 03           LXI  H,X03EA PUT START ADDR OF DEFAULT TABLE IN ACC                           !   1626
0BD6  06 00              MVI  B,H'00' CLEAR B REG   C REG IS 1ST ARG OFFSET                                1627
0BD8  09                 DAD  B COMPUTE ADDR OF 1ST-ARG VARIABLE LETTER TO BE REDEFINED                    1628
0BD9  73      ┌─>X0BD9   MOV  M,E STORE NEW DEFINITION SIZE IN TABLE                                       1629
0BDA  23      │          INX  H  ADVANCE PTR TO NEXT ENTRY IN TABLE                                    #   1630
0BDB  3D      │          DCR  A  DECREMENT CTR                                                         =   1631
0BDC  C2 D9 0B└──────────JNZ  X0BD9 IF NOT ZERO - MORE ENTRIES TO BE RE DEFINED YET                    BY  1632
0BDF  E1                 POP  H  RESTORE PTR TO STMT FROM STACK                                            1633
0BE0  C9                 RET  JMP TO CNTL-C CHK or ELSEWHERE                                           I   1634
---------------------------------------------------------------------------------------------------------------
0BE1  7E        X0BE1    MOV  A,M  FETCH CHAR INTO ACC              ┐ TEST CHAR POINTED                1635
0BE2  FE 41              CPI  A'A' } IF CHAR IS LESS THAN           │ TO BY HL                     A   1636
0BE4  D8                 RC        } AN 'A' RETURN WITH CARRY SET   │ IF CHAR NOT A-Z              X   1637
0BE5  FE 5B              CPI  A'[' } IF CHAR IS LARGER THAN A 'Z'   │ SET CARRY                    (RC 1638
0BE7  3F                 CMC       } RETURN WITH CARRY SET          │ RESET CARRY IF A-Z           ?   1639
0BE8  C9                 RET                                        ┘                              I RNC 1640
---------------------------------------------------------------------------------------------------------------
0BE9  D7        X0BE9    RST  2  SCAN STMT TO CHAR AFTER '('                                           W   1641
0BEA  CD 68 14  X0BEA    CALL X1468 EVALUATE STRING-OBTAIN 2 BYTE INTEGER VALUE OF ARG IN DE           M   1642
0BED  F0                 RP   RETURN WITH ARG IN DE ONLY IF VALUE IS ≥ 0                                   1643
0BEE  1E 05     X0BEE    MVI  E,H'05' } PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"                        1644 PRESENTLY SET FOR USR
0BF0  C3 B7 04           JMP  X04B7   }                                                                C7  1645
---------------------------------------------------------------------------------------------------------------
0BF3  2B        X0BF3    DCX  H  DECREMENT PTR TO STMT-BEING-PROCESSED, THIS IS DONE TO ENTER loop         1646 ASSEMBLE NXT GROUP OF DIGITS
0BF4  11 00 00  X0BF4    LXI  D,X0000 CLEAR DE REGISTER (TEMP STORAGE)                                     1647 INTO A 2 BYTE VALUE
0BF7  D7      ┌─>X0BF7   RST  2  POINT TO NXT NON-SPACE CHAR                                           W   1648
0BF8  D0      │          RNC   IF CHAR IS NOT A DIGIT, RETURN PACKED LINE # IN DE                      P   1649
0BF9  E5      │        ┌─PUSH H SAVE STMT PTR IN STACK                                                     1650
0BFA  F5      │        │┌PUSH PSW SAVE CHAR + STATUS FLAGS                                                 1651
0BFB  21 98 19│        ││LXI  H,X1998 LOAD H,L WITH 6552_10                                             !   1652
0BFE  E7      │        ││RST  4  COMPARE DE + HL   RZ HL=DE  RNC. HL≥DE  RC HL<DE                         1653
0BFF  DA A9 04│        ││JC   X04A9 LINE # > 65529, PRINT ERROR MESSAGE "SYNTAX ERROR"                 Z)  1654
0C02  62      │        ││MOV  H,D } SAVE # IN HL                                                            1655
0C03  6B      │        ││MOV  L,E }                                                                         1656
0C04  19      │        ││DAD  D  }                                                                          1657
0C05  29      │        ││DAD  H  } MULTIPLY BY 10_10                                                    )   1658
0C06  19      │        ││DAD  D  }                                                                          1659
0C07  29      │        ││DAD  H  }                                                                      )   1660
0C08  F1      │        │└POP  PSW RESTORE CHAR INTO ACC                                                    1661
0C09  D6 30   │        │ SUI  A'0' CONVERT DIGIT CHAR FROM ASCII TO HEX #                              V0  1662
0C0B  5F      │        │ MOV  E,A    } PUT THIS VALUE IN DE                                            _   1663
0C0C  16 00   │        │ MVI  D,H'00'}                                                                     1664
0C0E  19      │        │ DAD  D  ADD TO # IN HL                                                            1665
0C0F  EB      │        │ XCHG PUT # BACK IN DE                                                             1666
0C10  E1      │        └─POP  H  RESTORE STMT PTR                                                          1667
0C11  C3 F7 0B└──────────JMP  X0BF7 LOOK AT NXT CHAR IN STMT                                           C   1668
---------------------------------------------------------------------------------------------------------------
0C14  CA AB 05           JZ   X05AB IF NO ARGUMENT TO CLEAR, JUST DELETE ALL VARIABLES (PART OF "NEW")  J+  1669
0C17  CD EA 0B           CALL X0BEA CONVERT ARGUMENT FROM ASCII STRING TO NUMERICAL VALUE              M   1670
0C1A  2B                 DCX  H  BACK-UP PTR TO INSTRUCTION STRING                                     +   1671
0C1B  D7                 RST  2  TEST TERMINATOR CHAR AT THE END OF CLEAR INSTRUCTION                  W   1672
0C1C  C0                 RNZ   DO NOT PROCESS ARGUMENT TO SET STRING SPACE IF TERMINATOR NOT  A NULL or ':'  1673
0C1D  E5               ┌─PUSH H SAVE PTR TO INSTRUCTION ON STACK                                           1674
0C1E  2A A6 03         │ LHLD X03A6 LOAD HL WITH ADDR OF FIRST BYTE OF STRING AREA                     *&  1675
0C21  7D               │ MOV  A,L } COMPUTE NEW ADDR FOR END OF STRING AREA                                1676
0C22  93               │ SUB  E   } (DE) = (HL) - (DE)  -  DE HAS 2-BYTE VALUE FOR # OF                    1677
0C23  5F               │ MOV  E,A }                        BYTES TO RESERVE FOR CHAR                       1678
0C24  7C               │ MOV  A,H }                        STRINGS                                         1679
```



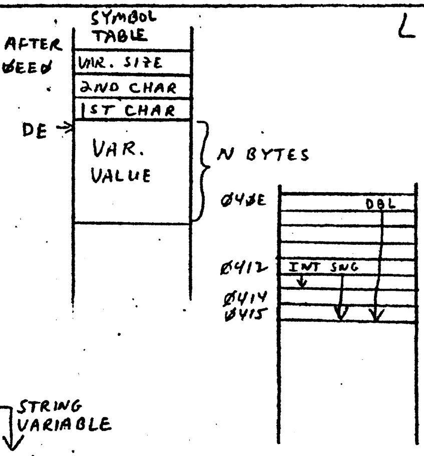
```
0925  9             SBB  D         } DE HAS NEW ADDR FOR
0926  57            MOV  D,A       } BYTE BELOW STRING AREA
0927  DA A9 04      JC   X04A9  IF # OF BYTES SPECIFIED BY ARGUMENT IS LARGER THAN       ADDR IN Ø3A6, PRINT "SYNTAX ERROR
092A  2A E1 03      LHLD X03E1  LOAD HL WITH ADDR OF BYTE AFTER END OF STMT AREA
092D  01 28 00      LXI  B,X0028} ADD 28H = 4Ø DECIMAL TO HL (ALLOW STACK SOME
0930  09            DAD  B      }                ROOM TO EXPAND
0931  E7            RST  4 TEST (DE) AGAINST (HL)
0932  D2 9E 04      JNC  X049E IF (HL)≥(DE); STACK WOULD OVERLAP STMT AREA, SO. PRINT    MESSAGE "OUT OF MEMORY"
0935  EB            XCHG       } STORE NEW ADDR (WHERE STACK WILL NOW START)
0936  22 D0 03      SHLD X03D0 } IN LOC Ø3D0
0939  E1            POP  H FETCH PTR TO INSTRUCTION FROM STACK
093A  C3 AB 05      JMP  X05AB DELETE ALL VARIABLES AFTER SETTING AMOUNT OF STRING SPACE   MOVE STACK BY CHANGING ADDR IN S
-------------------------------------------------------------------------------------------------------------
093D  CA A7 05      JZ   X05A7 IF 'RUN' WITH NO ARGUMENT, JMP TO 'NEW' SUBR -  J'START   PROGRAM AT FIRST STMT      RUN
0940  CC AB 05      CALL X05AB RESET FLAGS, THEN EXECUTE GOTO ADDR AFTER RUN TO  H+START  PROGRAM
0943  01 70 07      LXI  B,X0770} PUT A CNTL-C CHECK ADDR ON STACK FOR PSEUDO-GOTO
0946  C3 59 09      JMP  X0959  } INSTRUCTION                                  CY
-------------------------------------------------------------------------------------------------------------
0949  0E 03         MVI  C,H'03'} TEST IF 6 BYTES AVAILABLE BETWEEN END OF STACK +    END OF ARRAY TABLE         GOSUB
094B  CC 97 04      CALL X0497  }  (GOSUB ONLY NEEDS 5 BYTES)              H
094E  C1            POP  B REMOVE CNTL-C CHECK ADDR FROM STACK             A       AFTER Ø958
094F  E5            PUSH H SAVE PTR TO STMT ON STACK                                       STACK
0950  E5            PUSH H    } PUT LINE # OF CURRENT STMT ON STACK.                 PTR TO GOSUB INSTR
0951  2A 07 03      LHLD X0307} LEAVE PTR TO STMT IN HL                   *W          LINE # OF GOSUB
0954  E3            XTHL                                                              GOSUB TOKEN  <- STACK PTR
0955  16 8C         MVI  D,H'8C' PUT GOSUB TOKEN IN REG D                              CNTL-C CHECK ADDR
0957  D5            PUSH D } PUT REG D ON STACK (ONLY 1 BYTE PUT ON STACK)  U
0958  33            INX  SP }                                                3
0959  C5     X0959  PUSH B PUT CNTL-C CHECK ADDR BACK ON STACK [PERFORM GOTO E NXT]
-------------------------------------------------------------------------------------------------------------
095A  CD F3 08 X095A CALL X08F3 ASSEMBLE NXT GROUP OF CHAR'S INTO 2 BYTE VALUE (LINE # IN H DE)     GOTO
095D  CD 8E 09      CALL X098E AFTER LINE # EXTRACTED, SCAN REST OF STMT TIL NULL  M FOUND
0960  E5            PUSH H    SAVE PTR TO NXT STMT ON STACK - PTS. TO NULL AT END OF  STMT
0961  2A 07 03      LHLD X0307 FETCH LINE # OF CURRENT INSTRUCTION   RC  HL < DE   *W
0964  E7            RST  4 TEST IF CURRENT INSTR LINE # := LINE # AFTER GOTO  RNC HL ≥ DE
0965  F1            POP  H RESTORE PTR TO NXT STMT FROM STACK
0966  23            INX  H ADVANCE INSTRUCTION PTR TO POINT TO FIRST BYTE OF NXT INSTR AFTER GOTO
0967  DC 7E 05      CC   X057E IF LINE # OF GOTO IS LARGER THAN CURRENT LINE #, SCAN STMT   AREA STARTING AT CURRENT INSTR
096A  D4 7B 05      CNC  X057B IF LINE # OF GOTO IS LESS THAN CURRENT LINE #, SCAN STMT    AREA STARTING AT START OF STMT AREA
096D  60            MOV  H,B } PUT ADDR OF CHN ADDR OF STMT TO GOTO IN HL
096E  69            MOV  L,C }
096F  2B            DCX  H BACK-UP PTR BY 1 (ALWAYS POINT TO BYTE BEFORE INSTR WHEN + GOING TO CNTL-C CHECK)
0970  D8            RC   RETURN TO CNTL-C CHECK ONLY IF STMT WITH MATCHING LINE # X WAS FOUND IN Ø57E or Ø57B
-------------------------------------------------------------------------------------------------------------
0971  1E 08  X0971  MVI  E,H'08'} PRINT ERROR MESSAGE "UNDEFINED STATEMENT"
0973  C3 97 04      JMP  X0497  }                                          C7
-------------------------------------------------------------------------------------------------------------
0976  C0            RNZ  IGNORE RETURN IF NOT FOLLOWED BY A NULL or COLON   S                   RETURN
0977  16 FF         MVI  D,H'FF'                                                   IF A GOSUB IS ON STACK
0979  CC 57 04      CALL X0457 REMOVE A BYTE FROM STACK (AT ADDR SP+4)     MW                    STACK
097C  F9            SPHL PUT ADDR OF GOSUB LINE # IN SP                                     PTR TO GOSUB INSTR
097D  FE 8C         CPI  H'8C' IS BYTE A GOSUB TOKEN ?                     after SPHL SP -> LINE # OF GOSUB
097F  1E 03         MVI  E,H'03'} PRINT ERROR MESSAGE "RETURN WITHOUT GOSUB"         SP+4 -> GOSUB TOKEN
0981  C2 97 04      JNZ  X0497  } IF BYTE IS NOT A GOSUB TOKEN              B7              CNTL-C CHECK ADDR
0984  E1            POP  H     } FETCH LINE # OF GOSUB INSTR FROM STACK
0985  22 07 03      SHLD X0307 } STORE IT AT FLAG LOC FOR LINE # OF CURRENT INSTR  *W        SP -> Ø97C
0988  21 70 07      LXI  H,X0770} FETCH PTR TO GOSUB INSTR FROM STACK INTO HL, PUT CNTL-C   within Ø457
098B  E3            XTHL        } CHECK ADDR ON STACK, SCAN OVER REST OF GOSUB TIL NULL or ':' FOUND
-------------------------------------------------------------------------------------------------------------
DATA 098C  01 3A|0E  X098C  LXI  B,X0E3A  DATA - LXI B,XØE3A   REM - MVI C,ØØ       ELSE+REM + DATA
098F  00            NOP                                 ELSE
0990  06 00         MVI  B,H'00'                     B      C               DATA    B   C    SCAN TIL
0992  79   ->X0992  MOV  A,C } SWAP B+C REG'S  AFTER SWAP  DATA  ':'    NULL               Ø   3A   Colon or null Found
0993  48            MOV  C,B }                       REM   NULL   NULL      H    REM     Ø   Ø    NULL FOUND
0994  47            MOV  B,A } ACC HAS NULL          ELSE  NULL   NULL      G    ELSE    Ø   Ø    NULL FOUND
0995  7E   ->X0995  MOV  A,M FETCH NEXT CHAR FROM CURRENT INSTRUCTION  Ø98E  NULL  NULL   Ø98E    Ø   Ø    NULL FOUND
```

```
0996 B7          ORA  A } TEST CHAR + RETURN IF NULL (ANY OF 4 INSTR TYPES)   7
0997 C9          RZ     }                                                       H
0998 B8          CMP  B } TEST CHAR IF DATA STMT, RETURN WHEN COLON FOUND      8
0999 C8          RZ     }                                                       H
099A 23          INX  H  ADVANCE PTR TO NXT CHAR IN INSTRUCTION                #
099B FE 22       CPI  A'"'  } TEST CHAR, IF CHAR IS '"' (DATA INSTR) SCAN LINE TIL   "
099D CA 92 09    JZ   X0992 } ANOTHER '"' FOUND OR NULL                        J
09A0 D6 8A       SUI  H'8A' } JMP IF NXT CHAR IS NOT 'IF' TOKEN (8A)           V
09A2 C2 95 09    JNZ  X0995 } ACC=0                                            B
09A5 B4          CMP  B FORCE CARRY BIT SET IF 'IF' TOKEN FOUND (NOT WITHIN QUOTES)  8
09A6 8A          ADC  D ADD D + CARRY TO ACC
09A7 57          MOV  D,A PUT SUM BACK IN D, (ADD 1 TO D IF 'IF' TOKEN FOUND)  W
09A8 C3 95 09    JMP  X0995 CONTINUE LOOPING TIL NULL OR COLON FOUND           C
```

WHILE EXECUTING BASIC, THE CURRENT INSTRUCTION POINTER MUST BE PUSHED PAST CERTAIN INSTRUCTIONS TO LEAVE PTR AT ADDR BEFORE FIRST BYTE OF NXT EXECUTABLE INSTRUCTION

LET

```
09AB CD E0 0E  X09AB CALL X0EE0 FIND VARIABLE IN SYMBOL TABLE, ADD VARIABLE IF NOT FOUND, DE PT TO VALUE REGION
                                 HL PT TO STRING TO EVAL                              M
09AE CF              RST  1 } TEST THAT NEXT NON-SPACE CHAR IS '='                     O
09AF B9   DB 'B8'    CMP  B } IF NOT '=' CHAR, PRINT MESSAGE "SYNTAX ERROR"           8
09B0 EB              XCHG   } PUT PTR TO VARIABLE VALUE REGION IN LOC'S 03D3 + 03D4
09B1 22 D3 03        SHLD X03D3 } AFTER: HL PTS TO '=' IN INSTRUCTION } USED BY 'FOR'   S
09B4 EB              XCHG   }        DE PTS TO VARIABLE VALUE REGION } SEE 0775
09B5 D5        X09B5 PUSH D  STORE PTR TO VARIABLE REGION (IN SYMBOL TABLE) ON STACK  U
09B6 F7              RST  6  TEST VARIABLE TYPE (TEST LOC 03A4 + SET FLAGS)
09B7 F5              PUSH PSW SAVE FLAG STATUS + ADJUSTED TYPE CODE ON STACK
09B8 CD 52 0C        CALL X0C52 EVALUATE NEXT PART OF STRING - RESULT IN HOLDING AREA, TYPE FLAG IN 03A4  MR
09BB F1              POP  PSW FETCH ADJUSTED VARIABLE TYPE CODE FROM STACK
09BC E3   STORE VALUE IN  XTHL PUT PTR TO INSTRUCTION ON STACK, PUT PTR TO VARIABLE VALUE REGION IN HL
09BD C6 03 VARIABLE X09BD ADI H'03' READJUST VARIABLE TYPE CODE (COMPENSATE FOR RST 6)   F
09BF CD 90 11        CALL X1190 CONVERT VALUE IN HOLDING AREA TO FORMAT TO MATCH VARIABLE
09C2 CD 98 1B  INT,SNG,OR CALL X1B98 LOAD DE WITH ADDR OF LSB OF VALUE IN HOLDING AREA (DE=0412 OR =0411E)
09C5 E5        DBL   PUSH H SAVE PTR TO VARIABLE VALUE AREA (IN SYMBOL TABLE) ON STACK
09C6 C2 F4 09        JNZ  X09F4 JMP IF INT, SNG, OR DBL VARIABLE TYPE (FLAGS SET BY RST6 IN 1B98)  B
09C9 2A 12 04        LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF IDENTITY GROUP IN STRING FORMULA TABLE   STRING VARIABLE
09CC E5              PUSH H PUT ADDR OF CHAR COUNT BYTE ON STACK
09CD 23              INX  H ADVANCE PTR TO STRING FORMULA TABLE - OR STRING VAR IN SYMBOL OR ARRAY TABLES
09CE 5E              MOV  E,M } PUT PTR TO 1ST CHAR OF STRING VALUE (RESULT OF 0C52 AT 09B8)
09CF 23              INX  H   } IN DE                                                #
09D0 56              MOV  D,M                                                        V      RNG HL≥DE
09D1 21 A2 03        LXI  H,X03A2 LOAD HL WITH ADDR OF BYTE AFTER END OF LINE BUFFER  !"
09D4 E7              RST  4 TEST PTRS AGAINST EACH OTHER, TELLS WHERE STRING IS LOCATED
09D5 D2 E8 09        JNC  X09E8 JMP IF STRING IS IN LINE BUFR AREA (HL ≥ DE) - REMOVE IDENTITY GROUP FROM STRING STACK
09D8 2A D0 03        LHLD X03D0 LOAD HL WITH ADDR OF BYTE 1 LESS THAN START OF STRING AREA*}
09DB E7              RST  4 TEST PTRS AGAINST EACH OTHER
09DC D1              POP  D PUT ADDR OF CHAR COUNT BYTE IN DE                        G
09DD D2 F0 09        JNC  X09F0 JMP IF HL ≥ DE - STRING IS BELOW STACK - STRING IN INSTRUCTION AREA  R
09E0 2A E1 03        LHLD X03E1 LOAD HL WITH ADDR OF NXT-AVAILABLE BYTE IN STMT AREA (ALSO 2 BYTES BEFORE START OF SYMBOL & ARRAY TABLES
09E3 E7              RST  4 TEST PTRS AGAINST EACH OTHER
09E4 D2 F0 09        JNC  X09F0 JMP IF HL ≥ DE - IDENTITY GROUP IN STRING FORMULA TABLE   R   IDENTITY GROUP IN VARIABLE REGION
09E7 3E D1     POP D MVI A,H'D1' DUMMY PUT ADDR OF CHAR COUNT BYTE IN DE - FORCES REMOVAL OF IDENTITY GROUP FROM STRING FORMULA TABLE IN CALL 1396   Q
09E9 CD 96 13        CALL X1396 REMOVE 3 BYTE IDENTITY GROUP FROM FORMULA TABLE (IF MOST RECENT ENTRY)  H   SPECIAL CASE ie A$ = B$
09EC EB              XCHG PUT ADDR OF CHAR COUNT BYTE IN HL (SAME AS ADDR FROM 090C OR 09E8)   STRING FORMULA TABLE NOT INVOLVED
09ED CD CA 11        CALL X11CA PUT IDENTITY GROUP OF STRING ON STRING STACK (STRING FORMULA TABLE)  MZ   YET
09F0 CD 96 13  X09F0 CALL X1396 REMOVE IDENTITY GROUP FROM STRING FORMULA TABLE      M
09F3 E3              XTHL PUT PTR TO VARIABLE VALUE REGION IN HL, PUT AN ADDR ON STACK TO BALANCE POP D AT 09F7
09F4 CD 66 1B  X09F4 CALL X1B66 MOVE VALUE FROM HOLDING AREA TO AREA IN SYMBOL TABLE  H
09F7 D1              POP  D REMOVE 1 ADDR FROM STACK('FOR' - DE IS PTR TO VAR VALUE REGION)  Q
09F8 E1              POP  H FETCH PTR TO INSTRUCTION FROM STACK
09F9 C9              RET    END OF LET PROCESSING                                    I
```



ON

```
09FA CD 81 14        CALL X1481 EVALUATE ARGUMENT TO A 1 BYTE VALUE (STOP AT DELIMITER)   N
09FD 7E              MOV  A,M FETCH NXT CHAR FROM INSTRUCTION STRING (1 BYTE RETURNED IN REG E)
09FE 47              MOV  B,A SAVE CHAR IN REG B ('GOTO' OR 'GOSUB' TOKEN IN B)    G
09FF FE 8C           CPI  H'8C' IF CHAR IS A GOSUB TOKEN, JMP TO 0A07
```

```
0A01  CA 07 0A          JZ    X0A07 } IF CHAR IS A 'GOSUB' TOKEN, JMP TO ØAØ7                                  J
0A04  CF                RST   1 } SYNTAX CHECK FOR 'GOTO' TOKEN                                                O
0A05  88      DB '88' ~~ADC   B~~ }
0A06  2B                DCX   H BACK-UP PTR TO INSTRUCTION (RST 1 ADVANCES PTR 1 BYTE TOO FAR)
0A07  4B      X0A07     MOV   C,E PUT 1 BYTE OF EVALUATED ARGUMENT IN REG C                                    K
0A08  0C      X0A08     DCR   C DECREMENT ARGUMENT
0A09  7B                MOV   A,B PUT 'GOTO' OR 'GOSUB' TOKEN IN ACC
0A0A  CA 89 07          JZ    X0789 IF ARGUMENT HAS BEEN DECREMENTED TO ØØ, EXECUTE ROUTINE  9 TO  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  H  DELIMITER FOUND ie ',')
0A10  FE 2C             CPI   A',' } IF DELIMITER CHAR IS NOT A COMMA, THEN 'ON' INSTRUCTION IS            ,
0A12  C0                RNZ        } TERMINATED WITHOUT EXECUTING A 'GOTO' OR 'GOSUB'                      @
0A13  C3 08 0A          JMP   X0A08 LOOP TO PROCESS 'ON' INSTRUCTION                                           C
------------------------------------------------------------------------------------------------------------------
                                                                                                               IF
0A16  CD 52 0C          CALL  X0C52 EVALUATE EXPRESSION TO TRUE/FALSE VALUE (TRUE=-1, FALSE=Ø)                  MR
0A19  7E                MOV   A,M } TEST NEXT CHAR IN INSTRUCTION STRING
0A1A  FE 2C             CPI   A',' }                                                                           ,
0A1C  CC 01 07          CZ    X0701 IF CHAR IS A ',', USE RST 2 TO SCAN TO NEXT NON-SPACE CHAR (ALLOWS  LQ  COMMA BEFORE 'THEN')
0A1F  FE 88             CPI   H'88' } IF CHAR IS A 'GOTO' TOKEN, JMP TO ØA27
0A21  CA 27 0A          JZ    X0A27 }                                                                          J'
0A24  CF                RST   1 } OTHERWISE, CHAR MUST BE A 'THEN' TOKEN                                       O
0A25  AB      DB 'AB' ~~XRA   E~~ } PERFORM SYNTAX CHECK                                                       +
0A26  2B                DCX   H BACK-UP PTR TO INSTRUCTION (RST 1 ADVANCES PTR 1 BYTE TOO FAR)                +
0A27  E5      X0A27     PUSH  H SAVE PTR TO STMT ON STACK
0A28  CD 25 18          CALL  X1825 PERFORM SIGN TEST ON VALUE IN HOLDING AREA                                 H%
0A2B  E1                POP   H FETCH PTR TO STMT FROM STACK
0A2C  CA 36 0A          JZ    X0A36 IF ARG TO 'IF' INSTRUCTION IS FALSE, JMP TO ØA36                           J6
0A2F  D7      X0A2F     RST   2 SCAN TO NEXT NON-SPACE CHAR                                                    W
0A30  DA 5A 09          JC    X095A IF CHAR IS A DIGIT, TREAT IT AS A GOTO, + DIGIT IS 1ST CHAR OF A LINE  Z #
0A33  C3 38 07          JMP   X0738 OTHERWISE, JMP TO COMMAND PROCESSOR TO BRANCH TO ROUTINE TO EXECUTE  C8  NEXT INSTRUCTION
0A36  16 01   X0A36     MVI   D,H'01' SET 'IF' CTR TO 1, ALLOWS COMPLEX IF THEN (IF THEN ELSE) ELSE STMTS
0A38  CD 8C 09 X0A38    CALL  X098C SCAN INSTRUCTION TIL NULL OR COLON FOUND, IF COLON FOUND AFTER 'IF'  H TOKEN  , ADD 1 TO # IN D
0A3B  B7                ORA   A } IF NULL WAS FOUND, THEN NO MATCHING ELSE, RETURN TO CNTL-C  7 CHECK  + NEXT STMT IN PROGRAM
0A3C  C8                RZ      }                                                                              H
0A3D  D7                RST   2 SCAN TO NEXT NON-SPACE CHAR                                                    W
0A3E  FE 90             CPI   H'90' } IF CHAR AFTER COLON, IS NOT AN 'ELSE', SCAN INSTRUCTION FOR NEXT  COLON OR A NULL
0A40  C2 38 0A          JNZ   X0A38 }                                                                          B8
0A43  15                DCR   D DECREMENT 'IF' CTR
0A44  C2 38 0A          JNZ   X0A38 IF 'IF' CTR IS NOT ZERO, THEN INSTRUCTION MUST BE FURTHER SCANNED  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/  INSTRUCTION OR INSTRUCTION(S)
------------------------------------------------------------------------------------------------------------------
0A4A  2B      X0A4A     DCX   H BACK-UP PTR TO INSTRUCTION                                                     +   PART OF PRINT
0A4B  D7                RST   2 SCAN TO NEXT CHAR                                                              W
------------------------------------------------------------------------------------------------------------------
                                                                                                               PRINT
0A4C  CA 9B 0A          JZ    X0A9B IF COLON OR NULL AFTER PRINT, DO A CR + LF                                 J
0A4F  C8      X0A4F     RZ    IF NULL OR COLON LAST CHAR SCANNED, END OF PRINT EXECUTION                       H
0A50  FE AA             CPI   H'AA' } IF CHAR IS 'USING' TOKEN, DO FORMATTED OUTPUT                            *
0A52  CA 5E 15          JZ    X155E }                                                                          J^
0A55  FE A6             CPI   H'A6' } IF CHAR IS 'TAB(' TOKEN, DO TAB FUNCTION                                &
0A57  CA C5 0A          JZ    X0AC5 }                                                                          JE
0A5A  FE A9             CPI   H'A9' } IF CHAR IS 'SPC(' TOKEN, DO SPACE FUNCTION                              )
0A5C  CA C4 0A          JZ    X0AC4 }                                                                          JD
0A5F  E5                PUSH  H SAVE PTR TO INSTRUCTION STRING
0A60  FE 2C             CPI   A',' } IF CHAR IS ',' TOKEN, DO A 14 CHAR UNFORMATTED FIELD,                     ,
0A62  CA 90 0A          JZ    X0A90 }                                                                          J0
0A65  FE 3B             CPI   A';' } IF CHAR IS ';', RESTORE SCAN PTR TO HL, SCAN TO NEXT CHAR,  CONTINUE PROCESSING
0A67  CA E6 0A          JZ    X0AE6 }                                                                          J
0A6A  C1                POP   B REMOVE PTR TO INSTRUCTION FROM STACK INTO BC (JUST REMOVES ADDR  A  FROM STACK)
0A6B  CD 52 0C          CALL  X0C52 EVALUATE STRING (UP TO DELIMITER) TO VALUE IN HOLDING AREA                 MR
0A6E  E5                PUSH  H SAVE PTR TO INSTRUCTION ON STACK
0A6F  F7                RST   6 TEST FOR VARIABLE TYPE
0A70  CA 8C 0A          JZ    X0A8C IF CHAR STRING (OR STRING VARIABLE), PRINT CHAR STRING                     J
0A73  CD 81 21          CALL  X2181 CONVERT # IN HOLDING AREA TO EQUIVALENT TO EQUIVALENT CHAR  M STRING
0A76  CC FC 11          CALL  X11FC GENERATE IDENTITY GROUP FOR STRING, + PUT GROUP IN STRING FORMULA  M TABLE
```

```
0A79  2A 12 04          LHLD X0412 PUT ADDR OF CHAR COUNT BYTE OF IDENTITY GROUP IN HL
0A7C  3A 27 00          LDA  X0027 FETCH PRINT POSITION CTR
0A7F  86                ADD  M     ADD LENGTH OF STRING FOR CONVERTED #
0A80  FE 48             CPI  A'H'  } IF PRINT POSITION IS AT END OF LINE, PRINT CRLF + NULLS
0A82  D4 98 0A          CNC  X0A98 }
0A85  CD 44 12          CALL X1244 PRINT CHAR STRING OF CONVERTED # 
0A88  3E 20             MVI  A,A' ' } PRINT A SPACE AFTER #
0A8A  DF                RST  3     }
0A8B  B7                ORA  A     RESET ZERO BIT (ACC CONTAINS A 'SP')
0A8C  CC 44 12      X0A8C CZ  X1244 PRINT CHAR STRING
0A8F  E1                POP  H FETCH PTR TO INSTRUCTION FROM STACK
0A90  C3 4A 0A          JMP  X0A4A JMP TO CONTINUE SCANNING & PROCESSING PRINT INSTR
0A93  3E 00         X0A93 MVI  M,H'0D' PUT NULL AT END OF LINE BUFR
0A95  21 59 03          LXI  H,X0359 RESET HL TO POINT TO 1 LESS THAN START OF LINE BUFR
0A98  3E 00         X0A98 MVI  A,H'00' }
0A9A  32 27 00            STA  X0027   } PRINT CR
0A9D  DF                  RST  3       }
0A9E  3E 0A               MVI  A,H'0A' } PRINT LF
0AA0  DF                  RST  3       }
0AA1  3A 26 00      X0AA1 LDA  X0026 FETCH NULL CTR INTO ACC
0AA4  3D            X0AA4 DCR  A     DECREMENT NULL CTR
0AA5  32 27 00            STA  X0027 - EVENTUALLY PRINT POSITION CTR WILL BE ZEROED
0AA8  C8                  RZ       RETURN TO CNTL C CHECK OR ROUTINE THAT CALLED 0A98
0AA9  F5                  PUSH PSW STORE ACC (NULL CTR STORED ON STACK)
0AAA  AF                  XRA  A   PUT A NULL IN ACC
0AAB  DF                  RST  3   PRINT NULL
0AAC  F1                  POP  PSW RESTORE ACC (FETCH NULL CTR FROM STACK)
0AAD  C3 A4 0A            JMP  X0AA4 LOOP UNTIL ALL NULLS HAVE BEEN PRINTED
0AB0  3A 27 00      X0AB0 LDA  X0027 FETCH OUTPUT CHAR COUNT
0AB3  FE 3A               CPI  A'R' } IF AT PRINT POSITION 56 or LARGER, PRINT CRLF + NULLS
0AB5  D4 98 0A            CNC  X0A98 }
0AB8  D2 E6 0A            JNC  X0AE6 RESTORE SCAN PTR TO HL, SCAN TO NXT CHAR, CONTINUE PROCESSING
0ABB  D6 0E         X0ABB SUI  H'0E' } SUBTRACT 14. FROM FIELD LENGTH, REPEAT UNTIL RESULT
0ABD  D2 BB 0A            JNC  X0ABB } IS MINUS.
0AC0  2F                  CMA  COMPLEMENT REMAINDER, # IN ACC IS # OF SPACES-1 TO PRINT
0AC1  C3 DD 0A            JMP  X0ADD PRINT SPACES
0AC4  37            X0AC4 STC  CARRY SET FOR SPACE FUNCTION
0AC5  F5            X0AC5 PUSH PSW CARRY RESET FOR TAB FUNCTION, SAVE CARRY STATUS & TOKEN ON STACK
0AC6  CC 80 14            CALL X1480 FETCH # OF BLANKS TO PRINT FROM TAB OR SPACE ARGUMENT
0AC9  CF                  RST  1 } SYNTAX CHECK FOR ')' AFTER ARGUMENT
0ACA  29                  DB ')' 0A8-H }
0ACB  2B                  DCX  H  BACK UP PTR TO INSTRUCTION
0ACC  F1                  POP  PSW FETCH TOKEN FROM STACK
0ACD  FE A9               CPI  H'A9' TEST FOR SPC( TOKEN
0ACF  E5                  PUSH H SAVE PTR TO INSTRUCTION ON STACK
0AD0  3E FF               MVI  A,H'FF' PUT -1 IN ACC FOR SPC FUNCTION
0AD2  CA D9 0A            JZ   X0AD9 JMP IF SPC FUNCTION, ADD ARGUMENT (# of BLANKS) TO -1
0AD5  3A 27 00            LDA  X0027 FETCH SCAN PTR
0AD8  2F                  CMA  COMPLEMENT SCAN PTR TO DO SUBTRACT
0AD9  83            X0AD9 ADD  E ADD BLANK COUNT TO -1 (SPC) or NEGATED SCAN PTR (TAB)
0ADA  D2 E6 0A            JNC  X0AE6 IF RESULT IS NEGATIVE, SKIP PRINTING OF ANY BLANKS
0ADD  3C            X0ADD INR  A ADD 1 TO # OF BLANKS TO PRINT
0ADE  47                  MOV  D,A USE REG B TO STORE BLANK CTR
0ADF  3E 20               MVI  A,A' ' PUT 'SP' CHAR IN ACC
0AE1  DF            X0AE1 RST  3  PRINT 'SP'
0AE2  05                  DCR  B  DECREMENT SPACE CTR
0AE3  C2 E1 0A            JNZ  X0AE1 PRINT 'SP'S UNTIL CTR IS ZERO
0AE6  E1            X0AE6 POP  H FETCH PTR TO INSTRUCTION FROM STACK
0AE7  D7                  RST  2 SCAN TO NEXT CHAR
0AE8  C3 4F 0A            JMP  X0A4F JMP TO CONTINUE PROCESSING PRINT INSTRUCTION
```

```
*  1860
:' 1861
   1862
   1863
T  1964
MD 1865
>  1866
   1867
7  1868
LD 1869
   1870
CJ 1871
6  1872
!Y 1873
>  1874
2' 1975
_  1976
>  1877
_  1978
:& 1879
=  1880
2'WHEN 1881
H  1882
   1883
/  1884
-  1885
   1886
C& 1887
:' 1888
 8 1889
T  1890
RNG 1891
T  1892
R: 1893
 / TO 1894
C) 1895
7  1896
   1897
M  1898
O  1899
)  1900
+  1901
   1902
 ) 1903
   1904
>  1905
JY 1906
:' 1907
/  1908
   1909
R  1910
<  1911
G  1912
>  1913
-  1914
   1915
B  1916
   1917
W  1918
CO 1919
```

PRINT CR LF & NULLS

CLEAR PRINT POSITION CTR.

EXITING LOOP

14 CHAR UNFORMATTED FIELD

AFTER DOING CRLF NULLS

GET TO MULTIPLE OF 14 POSITION

SPACE FUNCTION FOR PRINT

TAB FUNCTION FOR PRINT

```
0AEB 3F        X0AEB  CMC                                                       ?     CHAR STRING
0AEC 52               MOV  D,D                                                  R
0AED 45               MOV  B,L                                                  E     "?REDO␣FROM␣START"
0AEE 44               MOV  B,H                                                  D
0AEF 4F               MOV  C,A                                                  O
0AF0 20               DATA A' '
0AF1 46               MOV  B,M                                                  F
0AF2 52               MOV  D,D                                                  R
0AF3 4F               MOV  C,A                                                  O
0AF4 4D               MOV  C,L                                                  M
0AF5 20               DATA A' '
0AF6 53               MOV  D,E                                                  S
0AF7 54               MOV  D,H                                                  T
0AF8 41               MOV  B,C                                                  A
0AF9 52               MOV  D,D                                                  R
0AFA D4 0D 0A         CNC  X0A0D                                                T
0AFD 00               NOP                                                             ERROR CAUSED BY INPUT OR READ
-------------------------------------------------------------------------------------------------------------
0AFE 3A 02 03  X0AFE  LDA  X0302 } TEST BRANCHING FLAG                          :R    VALUE NOT MATCHING VARIABLE TYPE
0B01 B7               ORA  A     } INPUT ØØ   READ AF                           7     OR ILLEGAL CHAR TYPED
0B02 C2 A3 04         JNZ  X04A3 IF 'READ' STMT, THEN PRINT "SYNTAX ERROR", USE LINE # OF  B#DATA STMT (COMMA WAS NOT WHERE EXPECTED)
0B05 C1               POP  B REMOVE 1 ADDR FROM STACK                           A
0B06 21 EB 0A         LXI  H,X0AEB LOAD HL WITH ADDR OF CHAR STRING "?REDO FROM START"  !
0B09 CD 41 12         CALL X1241 PRINT CHAR STRING                              MA
0B0C C3 E2 05         JMP  X05E2 RESTART INPUT INSTRUCTION                      C
-------------------------------------------------------------------------------------------------------------
0B0F FE 22            CPI  A'"' TEST 1ST NON-SPACE CHAR AFTER INPUT TOKEN       "     INPUT
0B11 3E 00            MVI  A,H'00' } CLEAR PRINT SUPPRESSION FLAG               >
0B13 32 A2 03         STA  X03A2   }                                            2"
0B16 C2 23 0B      ┌─ JNZ  X0B23 JMP IF CHAR IS NOT A '"'                      B#
0B19 CD FD 11      │  CALL X11FD GENERATE CHAR COUNT FOR STRING WITHIN QUOTES, CHAR COUNT IS IN C   M   ↓ PRINT MESSAGE WITHIN QUOTES
0B1C CF            │  RST  1 } SYNTAX CHECK FOR ';' AFTER CHAR STRING WITHIN QUOTES   O
0B1D 3B            │  DB';' DBY SP }                                            ;
0B1E E5    X0B1E   │ ┌PUSH H  SAVE PTR TO INSTRUCTION ON STACK
0B1F CD 44 12      │ │CALL X1244 PRINT CHAR STRING WITHIN QUOTES                MD
0B22 E1            │ └POP  H  FETCH PTR TO INSTRUCTION FROM STACK
0B23 E5    X0B23 <─┘ ┌PUSH H  SAVE PTR TO INSTRUCTION ON STACK
0B24 CD 03 11        │CALL X1103 TEST IF INPUT IS BEING USED AS A DIRECT COMMAND - ILLEGAL  M3-   PRINT ERROR MESSAGE-E
0B27 CD E6 05        │CALL X05E6 PRINT "?␣", INPUT CHAR STRING INTO LINE BUFR, LAST CHAR IS A NULL  M
0B2A 23              │INX  H  ADVANCE PTR TO POINT TO 1ST CHAR IN LINE BUFR    #
0B2B 7E              │MOV  A,M } TEST FIRST CHAR IN LINE BUFR
0B2C B7              │ORA  A   }                                               7
0B2D 2B              │DCX  H  BACK-UP PTR TO 1 BYTE BEFORE 1ST CHAR IN LINE BUFR  +
0B2E C1              └POP  B  REMOVE 1 ADDR FROM STACK                        A
0B2F CA FB 07  <─EXIT─ JZ   X07FB IF ONLY A CR WAS TYPED IN RESPONSE TO THE '?␣', EXECUTE 'END',  J   CONTINUE CAN BE USED TO RESTART
0B32 C5               PUSH B PUT ADDR BACK ON STACK (PTR TO INSTRUCTION)         E     PROGRAM AT
0B33 C3 3B 0B      ┌─ JMP  X0B3B                                                C:
-------------------------------------------------------------------------------------------------------------
0B36 E5            │  PUSH H  PUT PTR TO INSTRUCTION ON STACK                          READ
0B37 2A E7 03      │  LHLD X03E7 LOAD HL WITH ADDR IN PTR USED TO SCAN 'DATA' STMTS   •
0B3A F6 AF  <──────┘  ORI  H'AF' DUMMY XRA A } 'INPUT' STORES ØØ AT Ø3D2        /
0B3C 32 D2 03         STA  X03D2             } 'READ' STORES AF AT Ø3D2          2R
0B3F E3              ┌XTHL PUT PTR TO INSTRUCTION IN HL, PUT PTR TO STRING OF CHAR'S FOR VALUE ON STACK
0B40 01 CF 2C        │LXI B,X2CCF DUMMY RST1 DB',' SYNTAX CHECK FOR COMMA       O,
0B43 CD E0 0E        │CALL X0EE0 FIND VARIABLE OR ARRAY ELEMENT IN THE APPROPRIATE TABLE  M
0B46 E3              └XTHL PUT PTR TO STRING OF CHAR'S FOR VALUE IN HL, PUT PTR TO INSTRUCTION ON STACK
0B47 D5               PUSH D PUT PTR TO VARIABLE VALUE REGION ON STACK          U
0B48 7E               MOV  A,M } IS CHAR POINTED TO BY HL A COMMA? (COULD BE CHAR BEFORE START OF CHAR BUFR)
0B49 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)
0B4B CA 5B 0B      ┌─ JZ   X0B5B YES, CHAR IN DATA STMT IS A COMMA, JMP         J(
0B4E 3A D2 03      │  LDA  X03D2 } TEST BRANCHING FLAG                          :R
0B51 B7            │  ORA  A     } INPUT ØØ   READ AF                           7
0B52 C2 B5 0B      │  JNZ  X0BB5 JMP IF 'READ' INSTRUCTION BEING PROCESSED - FIND A 'DATA' STMT IN STMT AREA   B5
```

```
                                                  (INPUT NEED TO SATISFY ANOTHER VARIABLE)
0355  3F              MVI  A,A'?'  } PRINT '?' AS PROMPT CHAR                          >?
0357  DF              RST  3       }
0358  CD E6 05        CALL X05E6 PRINT '?␣', INPUT CHAR STRING INTO LINE BUFR, LAST CHAR IS A NULL   H
035B  F7       X0859  RST  6     TEST VARIABLE TYPE (LOC Ø3A4)              EVALUATE CHAR STRING TO VALUE
035C  F5              PUSH PSW   SAVE (TYPE-3) & FLAGS ON STACK             ARITHMETIC OR STRING TYPE
035D  C2 7A 08        JNZ  X087A JMP IF VARIABLE IS ARITHMETIC TYPE                    B
0860  D7              RST  2 SCAN TO CHAR (NON-SPACE) AFTER DELIMITER, CHAR IN ACC     W
0861  57              MOV  D,A } IF 1ST CHAR IS '"', USE THIS AS THE DELIMITER CHAR    W   STRING VARIABLE
0862  47              MOV  B,A } IN 12ØØ TO FIND THE END OF THE STRING                 G
0863  FE 22           CPI  A'"'                                                        "
0865  CA 60 08        JZ   X0860                                                       J
0868  16 3A           MVI  D,A':' } IF 1ST CHAR IS NOT '"', USE ',' & ':' AS DELIMITERS TO   :
086A  06 2C           MVI  B,A',' } FIND END OF STRING IN 12ØØ                         ,
086C  2B              DCX  H  BACK-UP PTR TO STRING - ALLOWS 1ST CHAR TO BE TREATED AS PART OF   +   STRING
0860  CD 00 12 X0860  CALL X1200 GENERATE A CHAR COUNT FOR IDENTITY GROUP OF STRING VALUE   M
0870  F1       X0870  POP  PSW PUT (TYPE-3) OF VARIABLE IN ACC
0871  EB              XCHG PUT PTR TO DATA STREAM IN DE
0872  21 81 08        LXI  H,X0881 } PUT PTR TO VARIABLE VALUE REGION IN HL            !
0875  E3              XTHL         } PUT ADDR Ø881 ON STACK, IMPLIED JMP TO LOOP UNTIL ALL VAR'S   SEPARATED BY COMMA'S HAVE BEEN PROCESSED
0876  D5              PUSH D PUT PTR TO DATA STREAM ON STACK                           U   RETURN TO Ø881
0877  C3 90 09        JMP  X0990 PUT NUMERIC VALUE, STORE IN VARIABLE VALUE REGION AFTER   C=   APPROPRIATE TYPE CONVERSION, IF NECESSARY
------------------------------------------------------------------------------------------
087A  D7       X087A  RST  2 SCAN TO FIRST CHAR OF STRING TO BE CONVERTED TO A VALUE   W   ARITHMETIC VARIABLE
087B  CD 43 20        CALL X2043 CONVERT STRING TO NUMERIC VALUE                       MC
087E  C3 70 08        JMP  X0870 STORE VALUE IN VARIABLE VALUE REGION, CONTINUE TO PROCESS 'READ' OR 'INPUT' STMT   C
------------------------------------------------------------------------------------------
0881  2B       X0881  DCX  H BACK-UP PTR TO DATA STREAM (IN 'DATA' STMT OR LINE BUFR)  +
0882  D7              RST  2 SCAN TO NEXT NON-SPACE CHAR                               W
0883  CA 88 08        JZ   X0888 JMP IF NULL OR COLON - END OF INPUT STREAM, CONTINUE TO   J   PROCESS 'READ' OR 'INPUT' TIL ALL VARIABLES HAVE BEEN SATISFIED
0886  FE 2C           CPI  A',' } IF CHAR IS NOT A COMMA, 'READ' -> "SYNTAX ERROR" MESSAGE,
0888  C2 FE 0A        JNZ  X0AFE } ILLEGAL CHAR IN INPUT STREAM  'INPUT' -> RESTART BASIC PROGRAM AFTER MESSAGE "?REDO␣FROM␣START"
088B  E3       X0888  XTHL PUT PTR TO INSTRUCTION IN HL, PUT PTR TO DATA STREAM ON STACK
088C  2B              DCX  H BACK-UP PTR TO INSTRUCTION                                +
088D  D7              RST  2 SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION (NULL, COLON, OR ANOTHER VARIABLE NAME)
088E  C2 41 08 <-- ANOTHER VARIABLE  JNZ X0841 JMP TO PROCESS ANOTHER VARIABLE         BA
0891  D1              POP  D PUT PTR TO INPUT STREAM IN DE                             Q   READ OR INPUT ALMOST DONE
0892  3A 02 03        LDA  X0302 } TEST BRANCHING FLAG                                 :R
0895  B7              ORA  A     } INPUT ØØ   READ AF                                  7
0896  EB   END OF 'READ' XCHG PUT PTR TO INPUT STREAM IN HL, PUT PTR TO INSTRUCTION IN DE
0897  C2 E6 07 <--    JNZ  X07E6 IF 'READ' INSTR, SAVE PTR TO 'DATA' IN Ø3E7 FLAG      B
089A  B6              ORA  M TEST IF LAST CHAR IN DATA STREAM (LINE BUFR) IS A NULL - IF NOT,   PRINT MESSAGE IN Ø89F
089B  21 A4 08        LXI  H,X08A4 LOAD HL WITH ADDR OF CHAR STRING "?EXTRA␣IGNORED"   !$
089E  D5              PUSH D PUT PTR TO INSTRUCTION ON STACK                           U
089F  C4 41 12        CNZ  X1241 PRINT CHAR STRING                                     DA
08A2  E1              POP  H PUT PTR TO INSTRUCTION IN HL
08A3  C9              RET  END OF PROCESSING 'INPUT' INSTRUCTION                       I
------------------------------------------------------------------------------------------
08A4  3F       X08A4  CMC                                                              ?   CHAR STRING
08A5  45              MOV  B,L                                                         E
08A6  58              MOV  E,B                                                         X   "?EXTRA␣IGNORED"
08A7  54              MOV  D,H                                                         T
08A8  52              MOV  D,D                                                         R
08A9  41              MOV  B,C                                                         A
08AA  20              DATA A' '
08AB  49              MOV  C,C                                                         I
08AC  47              MOV  B,A                                                         G
08AD  4E              MOV  C,M                                                         N
08AE  4F              MOV  C,A                                                         O
08AF  52              MOV  D,D                                                         R
08B0  45              MOV  B,L                                                         E
08B1  C4 00 0A        CNZ  X0A00                                                       D
08B4  00              NOP
------------------------------------------------------------------------------------------
08B5  CD 8C 09 X08B5  CALL X098C SCAN STMT POINTED TO BY HL TIL NULL FOUND             M   'READ' - FIND A 'DATA' STMT IN STMT AREA
```

```
0BB8  B7                ORA   A   TEST CHAR                                                      7
0BB9  C2 CE 00          JNZ   X0BCE IF CHAR IS A ':', THEN JMP TO TEST IF DATA INSTRUCTION, IF ONCHAR IS A NULL, LOOK AT NXT STMT
0BBC  23                INX   H  } TEST NXT 2 BYTES IN STMT AREA (CHAIN ADDR)                  #
0BBD  7E                MOV   A,M }
0BBF  23                INX   H   }                                                              #
0BBF  B6                ORA   M   }                                                              6
0BC0  1E 04             MVI   E,H'04' } IF CHAIN ADDR IS 0000, THEN NOT ENOUGH DATA STMTS TO
0BC2  CA B7 04          JZ    X04B7   } SATISFY ALL OF THE READ STMT'S, PRINT ERROR MESSAGE 'OUT OF DATA'   J7
0BC5  23                INX   H  } PUT LINE # OF STMT IN DE                                    #
0BC6  5E                MOV   E,M }                                                              ^
0BC7  23                INX   H   }                                                              #
0BC8  56                MOV   D,M }                                                              V
0BC9  EB                XCHG        } STORE LINE # AT 03CF IN FLAG AREA (USED FOR ERROR
0BCA  22 CF 03          SHLD  X03CF } MESSAGE, IF ERROR OCCURS                                 "O
0BCD  EB                XCHG
0BCE  D7       X0BCE    RST   2     SCAN TO NXT NON-SPACE CHAR IN STMT                           H
0BCF  FE 83             CPI   H'83' } TEST CHAR, IF NOT A 'DATA' TOKEN, SCAN OVER INSTRUCTION TO
0BD1  C2 B5 0B          JNZ   X0BB5 } THE NEXT COLON                                             B5
0BD4  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'   C(
----------------------------------------------------------------------------------------------------------------
0BD7  11 00 00          LXI   D,X0000 LOAD DE WITH 0000, FORCES EXIT FROM 0457 UPON FINDING MOST RECENT 'FOR' TOKEN ON THE STACK
0BDA  C4 E0 0E X0BDA    CNZ   X0EE0 IF NON-SPACE CHAR AFTER 'NEXT' TOKEN NOT A NULL OR ':', FIND VAR IN SYMBOL TABLE   D      NEXT
0BDD  22 03 03          SHLD  X0303 STORE PTR TO STMT IN 0303 (TEMP STORAGE IN FLAG AREA)      "S
0BE0  CD 57 04          CALL  X0457 TEST STACK CONTENTS TO SEE IF A 'FOR' GROUP OF BYTES IS ACTIVE & NEAR CURRENT SP POSITION   MW
0BE3  C2 AF 04          JNZ   X04AF IF 'FOR' TOKEN NOT FOUND AT SP+4, PRINT ERROR MESSAGE "NEXT WITHOUT FOR"   B/
0BE6  F9                SPHL  ADJUST STACK POINTER TO POINT TO SIGN-OF-STEP BYTE
0BE7  D5                PUSH  D PUT PTR TO VARIABLE (FROM FOR-GROUP OF BYTES ON STACK) ON 'STACK - NOT MATCHED BY POP' - ADJUSTS STACK PTR FOR 0C41   U
0BE8  7E                MOV   A,M FETCH SIGN-OF-STEP BYTE FROM FOR-GROUP
0BE9  23                INX   H ADVANCE PTR TO FOR-GROUP OF BYTES                                #
0BEA  F5                PUSH  PSW PUT SIGN-OF-STEP BYTE ON STACK
0BEB  D5                PUSH  D SAVE PTR TO VARIABLE (FROM FOR-GROUP OF BYTES ON STACK) ON STACK   U
0BEC  7E                MOV   A,M FETCH [(TYPE-3) OF STEP] BYTE FROM FOR GROUP
0BED  23                INX   H ADVANCE PTR TO FOR-GROUP OF BYTES                                #
0BEE  B7                ORA   A TEST TYPE OF STEP (INTEGER IS -1 IN ACC, SNG-PREC IS 1 IN ACC)   7
0BEF  FA 09 0C          JM    X0C09 JMP IF USING INTEGER ARITHMETIC
0BF2  CD 43 18          CALL  X1843 FETCH STEP SIZE FROM FOR-GROUP INTO HOLDING AREA            MC
0BF5  E3                XTHL  SAVE PTR TO FOR-GROUP ON STACK, PUT PTR TO VARIABLE VALUE REGION IN HL
0BF6  E5                PUSH  H PUT PTR TO VARIABLE BACK ON STACK
0BF7  CD 90 18          CALL  X1890 FETCH VAR VALUE FROM SYMBOL TABLE, ADD STEP TO VARIABLE VALUE   M
0BFA  E1                POP   H FETCH PTR TO VARIABLE FROM STACK
0BFB  CD 50 18          CALL  X1850 STORE THE RESULT OF THE ADDITION BACK INTO THE VARIABLE VALUE REGION IN SYMBOL TABLE   M}
0BFE  E1                POP   H FETCH PTR TO FOR-GROUP FROM STACK
0BFF  CD 54 18          CALL  X1854 FETCH LIMIT VALUE FROM FOR-GROUP INTO BCDE                  MT
0C02  E5                PUSH  H SAVE PTR TO FOR-GROUP ON STACK
0C03  CD A1 19          CALL  X1BA1 PERFORM SNG-PREC MAGNITUDE TEST, TEST IF SUM EXCEEDS LIMIT   M!
0C06  C3 32 0C          JMP   X0C32                                                              C2
0C09  23       X0C09    INX   H } ADVANCE PTR TO FOR-GROUP PAST 4 DUMMY BYTES PUT ON            #
0C0A  23                INX   H } STACK AT 076E & 076F - REMEMBER THE 17 BYTE ALIGNMENT         #
0C0B  23                INX   H }                                                                #
0C0C  23                INX   H }                                                                #
0C0D  4E                MOV   C,M } FETCH 2 BYTE INTEGER FORMAT STEP SIZE INTO BC               N
0C0E  23                INX   H   }                                                              #
0C0F  46                MOV   B,M }                                                              F
0C10  23                INX   H   }                                                              #
0C11  E3                XTHL  PUT PTR TO VARIABLE VALUE REGION IN HL, PUT PTR TO FOR-GROUP ON STACK
0C12  5E                MOV   E,M } FETCH VARIABLE VALUE FROM SYMBOL TABLE INTO DE              #
0C13  23                INX   H   }
0C14  56                MOV   D,M }                                                              V
0C15  E5                PUSH  H SAVE PTR TO VARIABLE VALUE REGION (HI-BYTE)
0C16  69                MOV   L,C } PUT STEP SIZE IN HL
0C17  60                MOV   H,B }
```

```
0C18  CD 55 10          CALL X1055 PERFORM INTEGER ADDITION, ADD STEP SIZE TO VARIABLE VALUE       MU
0C1B  3A A4 03          LDA  X03A4 } FETCH + TEST OPERATION-TYPE BYTE FROM FLAG AREA               $
0C1E  FE 04             CPI  H'04' }
0C20  CA 3E 19          JZ   X193E IF INTEGER ADD OVERFLOWED, RESULT IN HOLDING AREA WOULD BE SNG-PREC TYPE - PRINT ERROR MESSAGE "OVERFLOW"
0C23  EB                XCHG PUT RESULT OF ADDITION IN DE (1D55 PUTS RESULT IN HL + HOLDING AREA)
0C24  E1                POP  H FETCH PTR TO VARIABLE VALUE REGION FROM STACK
0C25  72                MOV  M,D } STORE THE RESULT OF THE ADDITION BACK INTO THE VARIABLE
0C26  2B                DCX  H   } VALUE REGION IN SYMBOL TABLE                                    +
0C27  73                MOV  M,E }
0C28  E1                POP  H FETCH PTR TO FOR-GROUP FROM STACK
0C29  D5                PUSH D PUT SUM ON STACK                                                   U
0C2A  5E                MOV  E,M } FETCH LIMIT VALUE FROM FOR-GROUP INTO DE                       ^
0C2B  23                INX  H   }                                                                #
0C2C  56                MOV  D,M }                                                                V
0C2D  23                INX  H   }                                                                #
0C2E  E3                XTHL PUT SUM IN HL, PUT PTR TO FOR-GROUP ON STACK
0C2F  CD CC 1B          CALL X1BCC PERFORM INTEGER MAGNITUDE TEST, TEST IF SUM EXCEEDS LIMIT      ML
0C32  E1         X0C32  POP  H FETCH PTR TO FOR-GROUP FROM STACK
0C33  C1                POP  B FETCH SIGN-OF-STEP BYTE FROM STACK                                 A
0C34  90                SUB  B TEST IF RESULT OF MAGNITUDE TEST + SIGN-OF-STEP INDICATE END OF 'FOR' INSTRUCTION
0C35  CD 54 1B          CALL X1B54 PUT 'FOR' LINE # IN DE, PUT 'FOR' TERMINATOR ADDRESS IN BC     MT
0C38  CA 44 0C          JZ   X0C44 JMP IF 'FOR' INSTRUCTION IS FINISHED                           JD
0C3B  EB                XCHG }  STORE 'FOR' LINE # IN FLAG AREA (Ø3D7 IS LOC FOR CURRENT
0C3C  22 07 03          SHLD X0307 }                            INSTRUCTION LINE #)              "W
0C3F  69                MOV  L,C } PUT 'FOR' TERMINATOR ADDRESS IN HL (BYTE AFTER TERMINATOR IS 1ST BYTE
0C40  60                MOV  H,B }                               OF NXT INSTRUCTION AFTER 'FOR')
0C41  C3 79 07          JMP  X0779 PUT 'FOR' TOKEN ON STACK, DO CNTL-C CHK, CONTINUE EXECUTING INSTR'S FOLLOWING 'FOR' INSTRUCTION   C
0C44  F9         X0C44  SPHL ELIMINATE 'FOR' GROUP FROM STACK BY MOVING STACK PTR
0C45  2A 03 03          LHLD X0303 FETCH PTR TO CURRENT INSTRUCTION FROM FLAG AREA (SEE Ø8DC)    *S
0C48  7E                MOV  A,M } TEST NXT CHAR IN 'NEXT' INSTRUCTION
0C49  FE 2C             CPI  A',' }                                                              ,
0C4B  C2 70 07          JNZ  X0770 IF CHAR IS NOT A ',', THEN 'NEXT INSTRUCTION IS COMPLETED, DO CNTL-C CHK   B
0C4E  D7                RST  2 SCAN TO NXT NON-SPACE CHAR IN INSTRUCTION                          W
0C4F  CD 0A 08          CALL X080A EXECUTE 'NEXT' FOR 2ND VARIABLE (NOTE: CALL PUTS ADDR ON STACK TO BALANCE USUAL CNTL-C CHK ADDR ON STACK FOR Ø457)   MZ
------------------------------------------------------------------------------------------------------------
0C52  2B         X0C52  DCX  H      DECREMENT PTR TO STMT                                         +
0C53  16 00      X0C53  MVI  D,H'00' SET PRECEDENCE BYTE TO ZERO - INITIALIZE TO EVALUATE STRING
0C55  D5         X0C55  PUSH D PUT PRECEDENCE BYTE OF PREVIOUS OPERATOR ON STACK.                U
0C56  0E 01             MVI  C,H'01' SET BYTE COUNT TO 1 FOR Ø487 ROUTINE - TEST FOR ROOM FOR 2 BYTES
0C58  CD 87 04          CALL X0487 TEST FOR ENOUGH AVAILABLE MEMORY                               M
0C5B  CD CB 0D          CALL X0DCB TEST LEADING CHAR - TAKE APPROPRIATE ACTION TO EVALUATE PART OF STRING   MK
0C5E  22 05 03          SHLD X0305 SAVE PTR TO STRING AT Ø3D5                                     "U
0C61  2A 05 03   X0C61  LHLD X0305 FETCH PTR TO STRING FROM STORAGE AT Ø3D5                       *U
0C64  C1         X0C64  POP  B      FETCH PRECEDENCE BYTE OF PREVIOUS OPERATOR FROM STACK         A
0C65  7E                MOV  A,M FETCH NEXT CHAR FROM STRING INTO ACC - SHOULD BE AN OPERATOR
0C66  16 00             MVI  D,H'00' SET OPERATOR REG TO Ø -OR- CLEAR UPPER BYTE OF PRECEDENCE REGISTER - SEE ØC91 + ØC9E
                                      (SEE ØC8B + ØC84)
0C68  D6 B7      X0C68  SUI  H'B7' SUBTRACT B7 FROM CHAR - 1 BEYOND LARGEST TOKEN FOR 2-ARGUMENT ARITHMETIC TOKEN   V7
0C6A  DA 82 0C          JC   X0C82 JMP IF ARITHMETIC TOKEN OR ASCII DELIMITER                     Z
0C6D  FE 03             CPI  H'03' } JMP IF TOKEN IS 1-ARGUMENT FUNCTION TOKEN - SGN (BA) TO MID (B4)
0C6F  D2 82 0C          JNC  X0C82 } OTHERWISE TOKEN IS >(B7), =(B8), <(B9)   ACC HAS Ø FOR >, 1 FOR =, 2 FOR <
0C72  FE 01             CPI  H'01' SET CARRY FLAG IF '>' TOKEN
0C74  17                RAL         MULTIPLY OFFSET BY 2
0C75  AA                XRA  D      COMBINE WITH PREVIOUS RELATIONAL OPERATOR OFFSET              *
0C76  BA                CMP  D      TEST IF SAME OPERATOR USED TWICE                              :
0C77  57                MOV  D,A    SAVE OFFSET OF ACCUMULATED OPERATORS                          W
0C78  DA A9 04          JC   X04A9 PRINT ERROR MESSAGE "SYNTAX ERROR" - IF OPERATOR USED TWICE    Z)
0C7B  22 CD 03          SHLD X03CD SAVE PTR TO LAST OPERATOR IN Ø3CD FLAG AREA                    "M
0C7E  D7                RST  2      SCAN TO NEXT NON-SPACE CHAR - COULD BE ANOTHER OPERATOR      W
0C7F  C3 68 0C          JMP  X0C68 LOOP UNTIL NON-RELATIONAL OPERATOR OR CHAR FOUND              C
0C82  7A         X0C82  MOV  A,D } TEST OPERATOR REGISTER
0C83  B7                ORA  A   }                                                                7
```

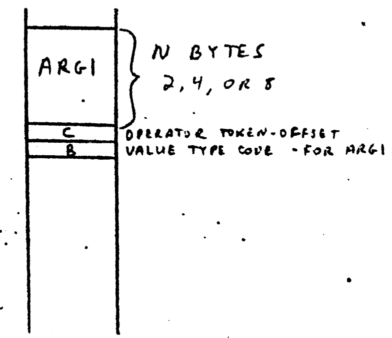
BIT WEIGHTS FOR RELATIONAL OPERATORS

- > Ø1
- = Ø2
- < Ø4

```
0C84  C2 12 00  <--        JNZ   X0D12 JMP IF EVALUATING RELATIONAL EXPRESSION (<, >, =>, etc)   B
0C87  7E                   MOV   A,M   PUT CHAR FROM STRING INTO ACC
0C88  22 C0 03             SHLD  X03C0 SAVE PTR TO STRING IN FLAG AREA                            "M
0C8B  D6 AE                SUI   H'AE' SUBTRACT AE FROM CHAR - SMALLEST TOKEN FOR ARITHMETIC TOKEN  V.
0C8D  D8        <-- DONE   RC          RETURN IF CHAR IS SMALLER THAN A '+' TOKEN - DELIMITER  X IN STRING } ONLY 9 TWO-ARG ARITHMETIC
0C8E  FE 09                CPI   H'09' } TEST CHAR + RETURN IF CHAR IS LARGER THAN A '\' TOKEN - DELIMITER IN STRING } OPERATORS
0C90  D0        <-- DONE   RNC         }                                                          P
0C91  5F                   MOV   E,A   SAVE ARITHMETIC TOKEN-OFFSET IN REG E                       -
0C92  3A A4 03             LDA   X03A4 } TEST TYPE OF COMPUTATION BEING USED TO EVALUATE THE STRING  :8
0C95  D6 03                SUI   H'03' }                                                          V
0C97  B3                   ORA   E     } PERFORM TEST - JMP IF TYPE IS Ø3 + ARITHMETIC OPERATOR IS +   3+
0C98  CA 2E 13  <--        JZ    X132E } - JMP TO PERFORM STRING VARIABLE CONCATENATION            J.
0C9B  21 71 00             LXI   H,X0071 LOAD HL WITH ADDR OF ARITHMETIC OPERATOR PRECEDENCE TABLE  !
0C9E  19                   DAD   D     COMPUTE ADDR OF PRECEDENCE BYTE FOR OPERATOR IN E
0C9F  7B                   MOV   A,D   PUT PREVIOUS-OPERATOR PRECEDENCE INTO ACC
0CA0  56                   MOV   D,M   PUT PRESENT-OPERATOR PRECEDENCE INTO REG D                  V
0CA1  BA                   CMP   D     COMPARE THE TWO PRECEDENCE OPERATORS                        :
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
0CA3  C5                   PUSH  B     PUT PREVIOUS-OPERATOR PRECEDENCE ONTO STACK                E   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        AREA) - WHEN DONE - IMPLIED JMP TO ØC61
0CA7  C5                   PUSH  B     } REASON FOR TEST AT ØC58                                   E
0CA8  7A                   MOV   A,D   PUT PRESENT-OPERATOR PRECEDENCE IN ACC
0CA9  FE 7F                CPI   H'7F' } TEST PRESENT-OPERATOR PRECEDENCE + JMP IF '↑' OPERATION
0CAB  CA F8 0C  --         JZ    X0CF8 }                                                          J
0CAE  FE 51                CPI   A'Q'  } TEST PRESENT-OPERATOR PRECEDENCE + JMP IF 'AND' or 'OR' OPERATION   Q
0CB0  DA 06 0D  <--        JC    X0D06 }                                                          Z
0CB3  E6 FE                ANI   H'FE' MASK THE LSB OF THE PRESENT-OPERATOR PRECEDENCE BYTE
0CB5  FE 7A                CPI   H'7A' } JMP IF 'MOD' or '\' OPERATION
0CB7  CA 06 0D  <--        JZ    X0D06 }                                                          J
0CBA  21 12 04      X0CBA  LXI   H,X0412 LOAD HL WITH ADDR OF INTEGER OR SNG-PREC VALUE IN HOLDING AREA
0CBD  B7                   ORA   A     CLEAR CARRY BIT - IF VALUE IS DBL-PREC, DOES NOT JC AT ØCD7   7
0CBE  3A A4 03      X0CBE  LDA   X03A4 LOAD ACC WITH TYPE-CODE OF VALUE IN HOLDING AREA           :$
0CC1  3D                   DCR   A     } DECREMENT ACC 3 TIMES - SETS FLAGS + TESTS TYPE-CODE       =
0CC2  3D                   DCR   A     }                                                          =
0CC3  3D                   DCR   A     }                                                          =
0CC4  CA 10 11  <--        JZ    X1110 JMP IF VALUE IS STRING TYPE                                J
0CC7  4E                   MOV   C,M   } FETCH 2 BYTES OF VALUE FROM MEMORY - PUT THEM ON THE STACK  N
0CC8  23                   INX   H     }                                                          #
0CC9  46                   MOV   B,M   }                                                          F
0CCA  C5                   PUSH  B     }                                                          E
0CCB  FA E6 0C  <--        JM    X0CE6 JMP IF VALUE IS INTEGER TYPE
0CCE  23                   INX   H     } FETCH 2 MORE BYTES OF VALUE FROM MEMORY -               #
0CCF  4E                   MOV   C,M   } PUT THEM ON THE STACK                                     N
0CD0  23                   INX   H     }                                                          #
0CD1  46                   MOV   B,M   }                                                          F
0CD2  C5                   PUSH  B     }                                                          E
0CD3  E2 E6 0C  <--        JPO   X0CE6 JMP IF VALUE IS SNG-PREC TYPE
0CD6  23                   INX   H     ADVANCE PTR TO VALUE                                       #
0CD7  DA DD 0C  --         JC    X0CDD JMP IF DBL-PREC - ENTRY AT ØCBE (see ØCBD)                 Z)
0CDA  21 0F 04             LXI   H,X040E LOAD HL WITH ADDR OF LSB OF DBL-PREC HOLDING AREA         !
0CDD  4E            X0CDD  MOV   C,M   } FETCH 4 MORE BYTES OF VALUE FROM MEMORY -               N
0CDE  23                   INX   H     } PUT THEM ON STACK                                         #
0CDF  46                   MOV   B,M   }                                                          F
0CE0  23                   INX   H     }                                                          #
0CE1  C5                   PUSH  B     }                                                          E
0CE2  4E                   MOV   C,M   }                                                          N
0CE3  23                   INX   H     }                                                          #
0CE4  46                   MOV   B,M   }                                                          F
0CE5  C5                   PUSH  B     }                                                          E
0CE6  DA 25 11      X0CE6  JC    X1125 JMP IF ENTERED AT ØCBE                                     Z%
```

```
0CE9  C6 03              ADI  H'03' CORRECT TYPE CODE TO TRUE VALUE  (see ØCC1 - ØCC3)          F
0CEB  4B                 MOV  C,E PUT ARITHMETIC TOKEN-OFFSET IN C  (see .ØC91 or ØD18)         K          */+-
0CEC  47                 MOV  B,A PUT VALUE TYPE CODE IN B                                       G
0CED  C5                 PUSH B   PUT TOKEN-OFFSET + TYPE-CODE ON STACK                          E
0CEE  01 20 00           LXI  B,X0D20 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                          E
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)  ME   Y^X
0CFB  CD 36 19           CALL X1D36 PUT THIS ARG ON THE STACK                                    M6
0CFE  01 DB 25           LXI  B,X25DB LOAD BC WITH THE ADDR OF EXPONENTIATION ROUTINE (USED AS 8%AN IMPLIED JMP)
0D01  16 7F              MVI  D,H'7F' SET PRECEDENCE REG (PREVIOUS OPERATOR) TO 7F
0D03  C3 F1 0C      <--- JMP  X0CF1                                                              C
0D06  D5         *X0D06  PUSH D SAVE CONTENTS OF DE ON THE STACK WHILE DOING THE NEXT ROUTINE    U          AND
0D07  CD 11 1C           CALL X1C11 CONVERT THE VALUE IN HOLDING AREA TO INTEGER FORMAT - ARG1   N          OR
0D0A  D1                 POP  D RESTORE CONTENTS OF DE FROM STACK                                Q          MOD
0D0B  E5                 PUSH H STORE INTEGER VALUE (FROM 1C11) ON THE STACK - ARG1                         \
0D0C  01 D2 0E           LXI  B,X0ED2 LOAD BC WITH THE ADDR OF ROUTINE TO HANDLE 'MOD' + '\' ARITHMETIC - OR 'AND' 'OR' FUNCTION
0D0F  C3 F1 0C      <--- JMP  X0CF1                                                              C
0D12  78         X0D12   MOV  A,B } TEST PRECEDENCE OF PREVIOUS OPERATOR                                    PRECEDENCE = 64  < = >
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   P
0D16  C5                 PUSH B SAVE PRECEDENCE OF PREVIOUS OPERATOR ON STACK                    E
0D17  D5                 PUSH D SAVE OFFSET OF ACCUMULATED OPERATORS ON STACK (see ØC77)         U
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 B ADDR ØD2D (see ØCBA - ØCF1)
0D23  2A 12 04           LHLD X0412 } SAVE ADDR THAT POINTS TO 1ST CHAR COUNT BYTE OF IDENTITY GROUP OF 1ST ARG. ON STACK (STRING TYPE ARG1)
0D26  E5                 PUSH H     }
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                                                              C
------------------------------------------------------------------------------------------------------------------------------
0D2D  C1         X0D20   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}                                                             2%
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 00           JZ   X0D60 }  (DBL-PREC */+-)                                           J
0D38  3A A4 03           LDA  X03A4 } IF ARG2 IS DBL-PREC, MOVE ARG2 TO TEMP STORAGE (Ø41F - Ø41F)  $3
0D3B  FE 08              CPI  H'08' } FETCH ARG1 FROM STACK INTO HOLDING AREA, CONVERT ARG1 TO DBL-PREC,
0D3D  CA 88 00           JZ   X0D88 } + USE DBL-PREC ARG'S                                       J
0D40  57                 MOV  D,A PUT ARG2 TYPE-CODE IN D                                        H
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 00           JZ   X0D9C }  (SNG-PREC */+-)                                           J
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 00           JNC  X0DA7 } ARG1 TO SNG-PREC, PUT ARG1 IN BCDE, ARG2 IN HOLDING AREA, USE SNG-PREC FCN'S
0D4D  CA 8A 1C           JZ   X1C8A 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               !   ARG1 + ARG2 ARE BOTH INTEGER
0D53  06 00              MVI  B,H'00' CLEAR B                                                        VALUES - USE INTEGER FCN'S
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             N
0D58  23                 INX  H   }                                                              #
0D59  46                 MOV  B,M }                                                              F
0D5A  D1                 POP  D  PUT ARG1 IN DE                                                  Q
0D5B  2A 12 04           LHLD X0412  PUT ARG2 IN HL                                              *
0D5E  C5                 PUSH B PUT ADDR OF FCN-ROUTINE ON STACK                                 E
                         RET  BRANCH TO FCN ROUTINE BY EXECUTING A RETURN
```

ARG1 FCN ARG2



```
005F  C9                  RET                                                                                  I
0060  CD 6F 1C   X0060    CALL  X1C6F CONVERT VALUE IN HOLDING AREA TO DBL-PREC (CDBL ROUTINE) - 2ND ARG       M
0063  CD 91 18            CALL  X1891 MOVE DBL-PREC VALUE FROM (Ø40E-Ø415) TO (Ø418-Ø41F) - 2ND ARG            M    DBL-PREC FCN DBL-PREC
0066  E1                  POP   H      } FETCH LO-4 BYTES      } FETCH 8-BYTE DBL-PREC #
0067  22 10 04            SHLD  X0410  } OF DBL-PREC # FROM    } FROM STACK INTO HOLDING AREA
006A  E1                  POP   H      } STACK INTO HOLDING                                                         ARG1 FCN ARG2
006B  22 0E 04            SHLD  X040E  } AREA                                                                          +
006E  C1         X006E    POP   B } FETCH SNG-PREC or upper 4 BYTES of DBL PREC                                A       -
006F  D1                  POP   D } # FROM STACK INTO BCDE                                                     Q       *
0070  CD 46 18            CALL  X1846 STORE SNG-PREC VALUE IN BCDE. INTO HOLDING AREA                          MF      ÷
0073  CD 6F 1C   X0073    CALL  X1C6F CONVERT VALUE IN HOLDING AREA TO DBL-PREC (CDBL ROUTINE)                 M
0076  21 FB 01            LXI   H,X01FB LOAD HL WITH ADDR OF DBL-PREC FUNCTION TABLE                           !    FCN'S WANT ARG1 IN
0079  3A A5 03   X0079    LDA   X03A5 LOAD ACC WITH OFFSET OF ARITHMETIC FCN TO PERFORM  } COMPUTE            %    Ø40E-Ø415 + ARG2 IN
007C  07                  RLC   MULTIPLY OFFSET BY 2 (ADDR TABLE - 2 BYTES PER ENTRY)    } ADDR OF                  Ø418-Ø41F
007D  C5                  PUSH  B SAVE BC ON STACK SINCE BC USED IN THE NEXT COMPUTATION } FCN ROUTINE     E
007E  4F                  MOV   C,A PUT TABLE-OFFSET IN C                                                      O
007F  06 00               MVI   B,H'00' CLEAR B
0081  09                  DAD   B COMPUTE ADDR OF ENTRY TO BE USED - PUT IN HL
0082  C1                  POP   B RESTORE BC FROM STACK                                                        A
0083  7E                  MOV   A,M } FETCH ADDR OF FCN-ROUTINE FROM TABLE
0084  23                  INX   H   }
0085  66                  MOV   H,M }
0086  6F                  MOV   L,A }
0087  E9                  PCHL        BRANCH TO FCN-ROUTINE
0088  C5         X0088    PUSH  B SAVE TYPE-CODE FOR 1ST ARG ON STACK                                          E    CONVERT ARG1 TO DBL-PREC
0089  CD 91 18            CALL  X1891 MOVE DBL-PREC VALUE FROM (Ø40E-Ø415) TO (Ø418-Ø41F) - 2ND ARG            M
008C  F1                  POP   PSW FETCH TYPE CODE FOR 1ST ARG FROM STACK INTO ACC
008D  32 A4 03            STA   X03A4 PUT TYPE-CODE IN FLAG AREA                                               2$
0090  FE 04               CPI   H'04' } IF 2ND ARG IS SNG-PREC, JMP TO Ø06E, FETCH 4-BYTES FROM STACK.              } FETCH 2 or 4 BYTES FOR ARG1
0092  CA 6E 00            JZ    X006E }                                                                        J    } FROM STACK - THEN CONVERT
0095  E1                  POP   H     } OTHERWISE, TYPE-CODE IS INTEGER TYPE, FETCH 2-BYTES FROM STACK              } ARG1 TO DBL-PREC - USE DBL-PREC
0096  22 12 04            SHLD  X0412 }                                                                             } FCN'S */+-
0099  C3 73 00            JMP   X0073 }                                                                        C
009C  CD 45 1C   X009C    CALL  X1C45 CONVERT VALUE IN HOLDING AREA TO SNG-PREC (CSGN ROUTINE) - 2ND ARG       ME   SNG-PREC
009F  C1                  POP   B } FETCH SNG-PREC # FROM STACK - 1ST ARG                                      A    ARG1 FCN ARG2
00A0  D1                  POP   D }                                                                            Q    FCN'S WANT ARG1 IN BCDE
00A1  21 05 02   X00A1    LXI   H,X0205 LOAD HL WITH ADDR OF SNG-PREC FUNCTION TABLE                           !       + ARG2 IN Ø412-Ø415
00A4  C3 79 00            JMP   X0079 COMPUTE ADDR FROM TABLE + BRANCH TO FCN TABLE                            C
00A7  E1         X00A7    POP   H FETCH 2-BYTE INTEGER # FROM STACK - 1ST ARG                                       CONVERT ARG1 TO SNG-PREC
00A8  CD 36 18            CALL  X1836 PUT SNG-PREC # (IN HOLDING AREA) ONTO STACK - 2ND ARG                    M6
00AB  CD 63 1C            CALL  X1C63 CONVERT # IN HL TO SNG-PREC - 1ST ARG                                    M
00AE  CD 51 18            CALL  X1851 PUT CONVERTED # IN BCDE (MOVE FROM HOLDING AREA) - 1ST ARG               MQ   ARG1 FCN ARG2
00B1  E1                  POP   H     } FETCH SNG-PREC # FROM STACK INTO
00B2  22 14 04            SHLD  X0414 } HOLDING AREA - 2ND ARG                                                      CONVERT ARGS TO SNG-PREC
00B5  E1                  POP   H     }                                                                             USE SNG-PREC FCN'S
00B6  22 12 04            SHLD  X0412 }
00B9  C3 A1 00            JMP   X00A1 COMPUTE ADDR FROM TABLE + BRANCH TO FCN TABLE                            C!
00BC  E5                  PUSH  H SAVE # IN HL ON STACK (2ND #)                                                     INTEGER DIVIDE
00BD  EB                  XCHG  MOVE # IN DE TO HL (1ST #)                                                          (# IN DE) ÷ (# IN HL)
00BE  CD 63 1C            CALL  X1C63 CONVERT # IN HL FROM INTEGER FORMAT TO SNG-PREC VALUE IN HOLDING AREA    M    (1ST #)   (2ND #)
00C1  E1                  POP   H FETCH 2ND # FROM STACK INTO HL
00C2  CD 36 18            CALL  X1836 PUT # IN HOLDING AREA ONTO STACK (1ST #)                                 M6
00C5  CD 63 1C            CALL  X1C63 CONVERT # IN HL TO SNG-PREC VALUE IN HOLDING AREA (2ND #)                M
00C8  C3 2C 1A            JMP   X1A2C JMP TO SNG-PREC DIVIDE ROUTINE                                           C,
00CB  D7         X00CB    RST   2 SCAN TO NXT CHAR    RC = DIGIT                                               W    PREFIX-CHAR EVALUATOR
00CC  DA 43 20            JC    X2043 JMP IF CHAR IS DIGIT - CONVERT STRING TO NUMERIC VALUE                   ZC   (EVALUATE / QUANTITY IN STRING TO
00CF  CD E1 08            CALL  X08E1 TEST CHAR   RNC A-Z                                                      M    A VALUE IN HOLDING AREA)
00D2  D2 0E 0E            JNC   X0E0E IF CHAR IS A-Z, STRING IS VARIABLE NAME, FETCH VARIABLE VALUE FROM SYMBOL TABLE INTO HOLDING AREA   R
00D5  FE AE               CPI   H'AE' } JMP IF CHAR IS '+' ('+' TOKEN IS AE) - LEADING '+' IS IGNORED          .
00D7  CA CB 00            JZ    X00CB }                                                                        JK
```

```
0DDA  FE           CPI   A'.'   } JMP. IF CHAR '.', NXT GROUP OF CHAR'S SHOULD BE DIGITS  .-   CONVERT STRING TO NUMERIC VALUE
0DDC  CA 43 20     JZ    X2043  }                                                           JC
0DDF  FE AF        CPI   H'AF'  } JMP IF CHAR '-', NEGATE NXT PART OF STRING TO BE EVALUATED
0DE1  CA 00 0E     JZ    X0E00  }                                                           J
0DE4  FE 22        CPI   A'"'   } JMP IF CHAR '"', EVALUATE STRING TO STRING-TYPE VALUE     ..
0DE6  CA F0 11     JZ    X11F0  }                                                           J
0DE9  FE AC        CPI   H'AC'  } JMP IF CHAR IS 'NOT' TOKEN, PERFORM 'NOT' FCN ON NXT PART OF STRING TO BE EVALUATED
0DEB  CA 99 0E     JZ    X0E99  }                                                           J
0DEF  FE A8        CPI   H'A8'  } JMP IF CHAR IS 'FN' TOKEN, EVALUATE USER FCN              (
0DF0  CA E8 10     JZ    X10E8  }                                                           J
0DF3  DE 8A        SUI   H'8A'    REDUCE TOKEN TO AN OFFSET                                 VJ
0DF5  D2 1C 0E     JNC   X0E1C    JMP IF CHAR IS 'SGN' TOKEN OR LARGER - TEST FOR SINGLE-ARG FUNCTION TOKEN   R
0DF8  CF     X0DF8 RST   1      } SYNTAX CHECK. CHAR MUST BE '('                            C
0DF9  28     DB'(' DATA A'('    }                                                           (
0DFA  CD 52 0C     CALL  X0C52  RECURSIVE CALL - EVALUATE STRING WITHIN PARENTHESES         MR
0DFD  CF           RST   1      } SYNTAX CHECK. CHAR MUST BE ')'                            O
0DFE  29     DB')' DAD   H      }                                                           )
0DFF  C9           RET          DONE                                                        I
-----------------------------------------------------------------------------------------------
0E00  16 7D  X0E00 MVI   D,H'7D' SET PRECEDENCE OF PREVIOUS OPERATOR TO 7D (LARGER THAN * / & LESS THAN ↑)
0E02  CD 55 0C     CALL  X0C55  EVALUATE NEXT PORTION OF STRING (UNTIL OPERATOR LESS THAN 7D IS FOUND) - THEN NEGATE THE FIRST VALUE)   MU
0E05  2A D5 03     LHLD  X03D5  LOAD HL WITH ADDR OF NEXT CHAR IN STRING TO BE PROCESSED   *U   ie  -A+B -> -(A)+B
0E08  E5           PUSH  H        SAVE PTR TO STRING ON STACK                               M        -A↑B -> -(A↑B)
0E09  CD DC 18     CALL  X18DC  NEGATE THE VALUE IN THE HOLDING AREA
0E0C  E1     X0E0C POP   H        FETCH PTR TO STRING FROM STACK
0E0D  C9           RET          DONE                                                        I
-----------------------------------------------------------------------------------------------
0E0E  CD E0 0E X0E0E CALL X0EE0 FIND VARIABLE IN SYMBOL TABLE - STMT PTR SCANS PAST NAME OF VARIABLE   FETCH VARIABLE VALUE FROM SYMBOL
0E11  E5     X0E11 PUSH  H  SAVE PTR TO STMT ON STACK                                               TABLE FOR USE IN A COMPUTATION
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 VARIABLE
0E16  F7           RST   6     TEST TYPE OF VARIABLE
0E17  C4 8B 1B     CNZ   X1B8B IF ARITHMETIC TYPE OF VARIABLE, CALL 1B8B, MOVE VALUE FROM SYMBOL TABLE TO HOLDING AREA
0E1A  E1           POP   H FETCH PTR TO STMT FROM STACK
0E1B  C9           RET   DONE                                                               I
-----------------------------------------------------------------------------------------------
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                                      O
0E20  C5           PUSH  B    SAVE ADJUSTED SINGLE-ARG FCN TOKEN ON STACK                   E
0E21  D7           RST   2    SCAN TO NEXT NON-SPACE CHAR IN STRING TO BE EVALUATED         W
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$                                  ZA
0E28  CF           RST   1    } SYNTAX CHECK FOR '('                                        O   LEFT$
0E29  28           DATA  A'(' }                                                             (   RIGHT$
0E2A  CD 52 0C     CALL  X0C52 EVALUATE 1ST ARG - SHOULD BE STRING-TYPE VALUE               MR  MID$
0E2D  CF           RST   1    } SYNTAX CHECK FOR ','                                        O
0E2E  2C           INR   L    }                                                             ,
0E2F  CD 88 1C     CALL  X1C88 PERFORM STRING VALUE CHECK ON VALUE IN HOLDING AREA          M
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 31 14     CALL  X1431 EVALUATE 2ND ARG. TO 1 BYTE INTEGER VALUE. VALUE IN E        M
0E3C  EB           XCHG } PUT VALUE FOR 2ND ARG ON STACK, PTR TO STRING TO BE EVALUATED IN DE
0E3D  E3           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   CV
-----------------------------------------------------------------------------------------------
0E41  CD F8 0D X0E41 CALL X0DF8 PERFORM SYNTAX CHECK FOR '(', EVALUATE ARG, PERFORM SYNTAX CHECK FOR ')'   M
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' }
```

```
0E48  DA 52 0E          JC    X0E52 JMP IF FCN IS SGN TO POS
0E4B  FE 1D             CPI   H'1D' TEST ADJUSTED TOKEN OFFSET
0E4D  E5                PUSH  H SAVE HL ON STACK SINCE HL MIGHT BE USED IF CC 1C45 at ØE4E
0E4E  DC 45 1C          CC    X1C45 CONVERT ARG IN HOLDING AREA TO SNG-PREC IF FCN IS SQR TO ATN E
0E51  E1                POP   H RESTORE HL FROM STACK (ADJUSTED TOKEN OFFSET)
0E52  11 0C 0E   X0E52  LXI   D,X0E0C } PUT ADDR ØEØC ON STACK FOR IMPLIED JMP
0E55  D5                PUSH  D       } - AFTER EXECUTING FCN ROUTINE, PUT PTR TO STRING IN HL, THEN RETURN TO ØC5E
0E56  01 3B 00   X0E56  LXI   B,X003B LOAD BC WITH ADDR OF START OF SINGLE-ARG FCN TABLE
0E59  09         X0E59  DAD   B  COMPUTE ADDR OF FCN ROUTINE ADDR IN TABLE - PUT IN HL
0E5A  4E                MOV   C,M }
0E5B  23                INX   H   } FETCH ADDR OF FCN ROUTINE INTO HL
0E5C  66                MOV   H,M }
0E5D  69                MOV   L,C }
0E5E  E9                PCHL     BRANCH TO ROUTINE TO PERFORM FCN
-----------------------------------------------------------------------------------------------
0E5F  CD 77 13   X0E5F  CALL  X1377 REMOVE IDENTITY GROUP OF 2ND ARG FROM FORMULA TABLE, REMOVE STRING FROM STRING AREA
0E62  7E                MOV   A,M  PUT CHAR COUNT BYTE OF 2ND ARG IN ACC
0E63  23                INX   H   }
0E64  4E                MOV   C,M } PUT ADDR OF 1ST CHAR OF 2ND ARG STRING IN BC
0E65  23                INX   H   }
0E66  46                MOV   B,M }
0E67  D1                POP   D    PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN DE (see ØD23-ØD26)
0E68  C5                PUSH  B    PUT PTR TO 1ST CHAR OF 2ND ARG STRING ON STACK
0E69  F5                PUSH  PSW  PUT CHAR COUNT OF 2ND ARG ON STACK
0E6A  CD 7E 13          CALL  X137E REMOVE IDENTITY GROUP OF 1ST ARG FROM FORMULA TABLE, REMOVE STRING FROM STRING AREA
0E6D  D1                POP   D    PUT CHAR COUNT BYTE OF 2ND ARG IN D
0E6E  5E                MOV   E,M  PUT CHAR COUNT BYTE OF 1ST ARG IN E (HL POINTS TO CHAR COUNT BYTE DUE TO CALL 137E AT ØE6A)
0E6F  23                INX   H   }
0E70  4E                MOV   C,M } PUT PTR TO 1ST CHAR OF 1ST ARG IN BC
0E71  23                INX   H   }
0E72  46                MOV   B,M }
0E73  E1                POP   H    PUT PTR TO 1ST CHAR OF 2ND ARG IN HL
0E74  7B         X0E74  MOV   A,E } TEST IF BOTH CHAR CTR'S ARE Ø
0E75  B2                ORA   D   }
0E76  C8   <-ARG1=ARG2- RZ         EXIT IF BOTH CHAR CTR'S ARE Ø
0E77  7A                MOV   A,D  PUT CHAR CTR OF 2ND ARG IN ACC
0E78  D6 01  ARG1>ARG2  SUI   H'01' } RETURN IF 2ND ARG CHAR COUNT IS ZERO
0E7A  D8   <-IN LENGTH- RC          }
0E7B  AF                XRA   A  CLEAR ACC
0E7C  BB                CMP   E }
0E7D  3C     ARG1<ARG2  INR   A }
0E7E  D0   <-IN LENGTH- RNC     } RETURN IF 1ST ARG CHAR COUNT IS ZERO
0E7F  15                DCR   D } DECREMENT CHAR CTR'S FOR BOTH ARGS
0E80  1D                DCR   E }
0E81  0A                LDAX  B FETCH A CHAR FROM 1ST ARG STRING INTO ACC
0E82  BE                CMP   M COMPARE AGAINST A CHAR IN 2ND ARG STRING
0E83  23                INX   H ADVANCE CHAR POSITION PTR TO 2ND ARG STRING
0E84  03     STRINGS    INX   B ADVANCE CHAR POSITION PTR TO 1ST ARG STRING
0E85  CA 74 0E  STILL MATCH  JZ  X0E74 IF THE CHAR'S MATCH, LOOP TO TEST THE NEXT PAIR OF CHAR'S IN THE STRINGS FOR A MATCH
0E88  3F                CMC      COMPLEMENT CARRY BIT, CARRY 1ST > 2ND , NO CARRY 1ST < 2ND   TO ? MATCH RESULTS IN TABLE
0E89  C3 F1 1A          JMP   X1AF1 PUT RESULT IN ACC, IF 1ST > 2ND ACC=-1 CARRY SET, IF 1ST < 2ND ACC=1, CARRY NOT SET
-----------------------------------------------------------------------------------------------
0E8C  3C         X0E8C  INR   A } ADJUST RESULT OF MAGNITUDE   >  =  <
0E8D  8F                ADC   A } TEST                         -1  Ø  1
0E8E  C1                POP   B FETCH RELATIONAL OPERATOR MASK FROM STACK (see ØC72-ØC77)   Øc  1nc 2nc
0E8F  A0                ANA   B AND MASK WITH RESULT OF TEST                                1   2   4
0E90  CE FF             ADI   H'FF' } ADJUST RESULT       {ACC=Ø IF FALSE
0E92  9F                SBB   A     } TRUE = -1 FALSE = Ø CARRY SET IF TRUE  {ACC=1 IF TRUE
0E93  CD 1E 1B          CALL  X1B1E CONVERT TRUE OR FALSE VALUE TO A 2 BYTE INTEGER - STORE IN HOLDING AREA
0E96  C3 61 0C          JMP   X0C61 JMP TO ØC61 TO PROCESS PRESENT OPERATOR
-----------------------------------------------------------------------------------------------
0E99  16 5A      X0E99  MVI   D,A'Z' SET PRECEDENCE OF PREVIOUS OPERATOR TO 5A (LARGER THAN 'AND', LESS THAN + -)
0E9B  CD 55 0C          CALL  X0C55 EVALUATE NEXT PORTION OF STRING (UNTIL OPERATOR LESS THAN 5A IS FOUND - MULT THEN 'NOT' THE FIRST VALUE)
```

COMPARISON OF 2 STRINGS FOR AN IF STMT

RESULT IN ACC

	ACC	CARRY
ARG1 = ARG2	Ø	Ø
ARG1 > ARG2	-1	1
ARG1 < ARG2	1	Ø

NOTE: THESE MATCH CONDITIONS DOCUMENTED IN ØE7C

AFTER DOING ARG1 COMPARE ARG2 (MAGNITUDE COMPARISON) TEST IF TRUE OR FALSE QUANTITY IN STRING BEING EVALUATED

RESULT OF COMPARISON TEST ARG1 ? ARG2
> ACC = -1
= ACC = Ø
< ACC = 1

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).                            A 2468
0EAB  C3 61 0C        JMP  X0C61 JMP TO ØC61 TO PROCESS PRESENT OPERATOR                      C 2469
-------------------------------------------------------------------------------------------------
0EAE  3D        X0EAE DCR  A } DECREMENT ACC 3 TIMES                                          = 2470 END OF RST 6
0EAF  3D              DCR  A }                                                                = 2471 RZ - CHAR STRING VARIABLE
0EB0  3D              DCR  A } SETS ZERO FLAG, DOES NOT AFFECT CARRY FLAG                     = 2472 RNC - DOUBLE PRECISION
0EB1  C9              RET                                                                     I 2473 RP - SNG PRECISION  RM - INTEGER
-------------------------------------------------------------------------------------------------
0EB2  C5        X0EB2 PUSH B    SAVE PREVIOUS OPERATOR PRECEDENCE ON STACK                    E 2474
0EB3  CD 11 1C        CALL X1C11 CONVERT ARG2 TO INTEGER FORMAT - IN HL + HOLDING AREA         M 2475
0EB6  F1              POP  PSW  FETCH PREVIOUS OPERATOR PRECEDENCE INTO ACC FROM STACK          2476
0EB7  D1              POP  D    FETCH ARG1 FROM STACK (see ØD0B)                            Q 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 }                                                            J1 2479
0EBD  FE 7B           CPI  H'7B' } IF PREVIOUS OPERATOR IS 7B, PERFORM ARG1 \ ARG2            2480     ↑ previous op   ↑ present operator
0EBF  CA CC 1D        JZ   X1DCC }                                                            JL 2481
0EC2  FE 46           CPI  A'F' TEST PREVIOUS OPERATOR PRECEDENCE BYTE                       F 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                                   JO 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)          CV 2489
0ECF  B5        X0ECF ORA  L  } OR LO-ORDER BYTES OF ARG1 + ARG2                              5 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  }                                                               Z 2493
0ED3  C3 D6 10        JMP  X10D6 STORE RESULT IN HOLDING AREA (INTEGER FORMAT VALUE)          CV 2494
-------------------------------------------------------------------------------------------------
0ED6  2B        X0ED6 DCX  H                                                                  + 2495   PART OF DIM
0ED7  D7              RST  2 } TEST NXT CHAR, IF NULL OR ':' RETURN                           H 2496   STRING SCAN LOOP
0ED8  C8              RZ     }                                                                H 2497   FOR NEXT ARRAY TO DIM
0ED9  CF              RST  1 } OTHERWISE NXT CHAR MUST BE A COMMA                            O 2498
0EDA  2C              INR  L DB ',' } SYNTAX ERROR IF NOT A COMMA                            , 2499
-------------------------------------------------------------------------------------------------
0EDB  01 D6 0E        LXI  B,X0ED6 } PUT ADDR ON STACK (PROVIDES LOOPING BY USE OF RETURN - IMPLIED JMP)  V 2500   DIM
0EDE  C5              PUSH B                                                                  E 2501
0EDF  F6 AF           ORI  H'AF' DUMMY XRA A } REMOTE ENTRY CLEAR ACC + LOC Ø3A3               / 2502
0EE1  32 A3 03        STA  X03A3 }                                                            2B 2503
0EE4  46              MOV  B,M  PUT FIRST CHAR OF NAME IN B REG                               F 2504
0EE5  CD E1 08  X0EE5 CALL X08E1 TEST CHAR - PUTS CHAR IN ACC FROM MEMORY FOR THE TEST        M 2505
0EE8  DA A9 04        JC   X04A9 PRINT MESSAGE "SYNTAX ERROR" IF NOT A-Z (FIRST CHAR          Z 2506 OF MATRIX NAME)
0EEB  AF              XRA  A  } CLEAR ACC                                                     / 2507
0EEC  4F              MOV  C,A } + C REG (2ND CHAR IS A NULL IF NOT A-Z OR A DIGIT)           O 2508
0EED  D7              RST  2 POINT TO NXT NON-SPACE CHAR (2ND CHAR OF NAME)                   W 2509
0EEE  DA F7 0E        JC   X0EF7 JMP IF 2ND CHAR IS A DIGIT  {IF ANY}                         Z 2510
0EF1  CD E1 08        CALL X08E1 TEST CHAR (POINTED TO BY HL)                                 M 2511
0EF4  DA 02 0F        JC   X0F02 JMP IF NOT A-Z                                               Z 2512
0EF7  4F        X0EF7 MOV  C,A PUT 2ND CHAR OF NAME IN C                                      O 2513
0EF8  D7        X0EF8 RST  2 ADVANCE PTR TO NXT NON-SPACE CHAR                                W 2514
0EF9  DA F8 0E        JC   X0EF8 JMP IF NXT CHAR IS DIGIT } LOOP ADVANCES HL PAST            Z 2515
0EFC  CD E1 08        CALL X08E1 TEST CHAR                 } DIGITS + A-Z CHAR               M 2516
0EFF  D2 F8 0E        JNC  X0EF8 JMP IF A-Z                (IGNORE ANY OVER 2 CHAR IN       R 2517 ARRAY NAME)
0F02  11 29 0F  X0F02 LXI  D,X0F29 } PUT ØF29 ON STACK (TO BE USED WITH RZ OR RET AS A JMP)   2518
0F05  D5              PUSH D                                                                  U 2519
```

STRNG EVALUATOR 'ERASE' & 'LET' ENTRY POINT

DIGIT

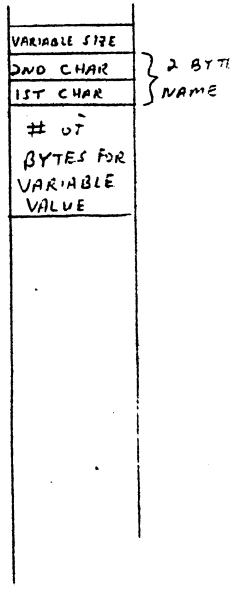
NOT A-Z

↑ DIGIT

A-Z

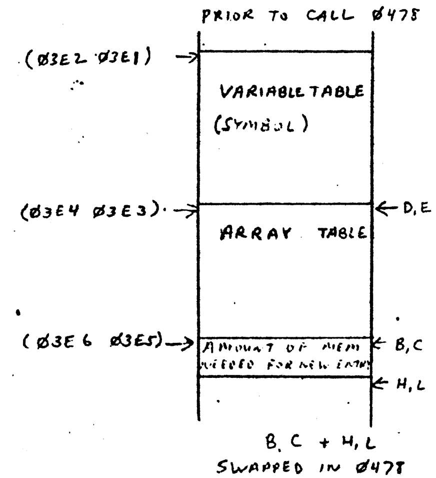
```
                        NXT TEST FOR ARRAY DEFINITION
0F06  16 02             MVI  D,H'02'  } SET D=2  IF
0F08  FE 25             CPI  A'%'     } INTEGER TYPE
0F0A  C8                RZ   (JMPS TO ØF29)
0F0B  14                INR  D        } SET D=3 IF STRING TYPE
0F0C  FE 24             CPI  A'$'     }
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          XØF29 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 Ø301 TO ØØ - TEMP STORAGE - ERASE SETS Ø3D1 TO 1
                                                                  FOR SETS Ø301 TO 64
0F31  3D                DCR  A
0F32  CA E1 0F    ERASE ←JZ   XØFE1 JMP IF ERASE COMMAND IS BEING PROCESSED
0F35  86                ADD  M  ADD CHAR '(' TO ACC
0F36  D6 27   SCAN ARRAY TABLE SUI  A'' SUB ')' FROM ACC (1 LESS THAN '(')
0F38  CA B8 0F          JZ   XØFB8 JMP IF CHAR IS '('
0F3B  AF                XRA  A  } CLEAR LOC Ø3D1
0F3C  32 D1 03          STA  X03D1 }
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 1ST BYTE OF SYMBOL TABLE
0F44  EB          XØF44 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   XØF69 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  XØF60 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  XØF60 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   XØFA3 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 }
```

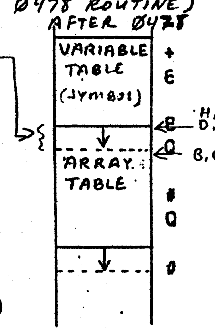
	2520
%	2521
H	2522
	2523
$	2524
H	2525
	2526
!	2527
H	2528
	2529
#	2530
H	2531
	2532
VA	2533
	2534
-	2535
	2536
	2537
!	2538
	2539
V	2540
	2541
+	2542
I	2543
	2544
2$	2545
H	2546
:Q	2547
=	2548
J	2549
	2550
V'	2551
J:	2552
/	2553
2Q	2554
	2555
U	2556
	2557
	2558
	2559
	2560
	2561
J	2562
	2563
	2564
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

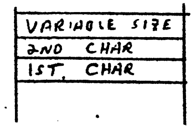


```
0F66  C3 44 0F                 JMP  X0F44   ENTER HERE IF SYMBOL_NAME (+ SIZE DEFN) NOT FOUND IN CD      SYMBOL TABLE
0F69  7C VARIABLE NOT  X0F69   MOV  A,H     PUT VARIABLE SIZE IN ACC (VARIABLE IN DIM STMT)              BEFORE ØED6   AFTER PTR TO STMT
0F6A  F1 FOUND IN TABLE        POP  H    }  LEAVE PTR TO STMT ON STACK                                  PTR TO STMT
0F6C  E3 ADD NEW ENTRY         XTHL      }  REMOVE IMPLIED JMP ADDR ØED6 FROM STACK, LEAVE IT IN HL
                                            - or return addr if entered at ØEEØ
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 ØEØØ - FIND VARIABLE IN SYMBOL      C
0F72  D1                       POP  D       } TABLE + PUT VARIABLE VALUE IN HOLDING AREA
0F73  CA A6 0F <-              JZ   X0FA6.                                                               J&
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)  E
0F7A  4F                       MOV  C,A   } SET UP BC   B IS ØØ  C IS VARIABLE-TYPE SIZE                  C
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                        A
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               M
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 (ARRAY) TABLE (ACCOMPLISHED BY
0F90  69                       MOV  L,C                                                   Ø478 ROUTINE)
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)                          6
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 D,E (D IS 1ST  E IS 2ND CHAR)              Q
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')                I
------------------------------------------------------------------------------------------------------------------------------
0FA6  32 15 04         X0FA6   STA  X0415   STORE ØØ AT LOC Ø415 (IF FLOATING-PT VARIABLE NOT FOUND IN TABLE - VALUE IS ØØ) - ENTRY NOT PUT IN TABLE YET.
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 ØCSE   B
0FB3  21 47 04                 LXI  H,X0447 } STORE ADDR OF NOP (PTR TO BYTE - SAYS STRING VARIABLE IS Ø IN LENGTH)   !G
0FB6  22 12 04                 SHLD X0412  } AT Ø412-Ø413
0FB9  E1                       POP  H       PUT PTR TO STRING IN HL (PUT ON STACK AT ØF6B)              I
0FBA  C9                       RET          - RETURN TO ØCSE

   IF VARIABLE NOT IN TABLE PUT ØØ OR NULL STRING PTR IN HOLDING AREA, DON'T
   BOTHER CREATING THE VARIABLE IN THE SYMBOL TABLE
------------------------------------------------------------------------------------------------------------------------------
0FBB  E5               X0FBB   PUSH H     } PUT VARIABLE TYPE (LOC Ø3A4) ON STACK } SAVE Ø3A3 + Ø3A4 ON STACK     ARRAY TABLE SEARCH
0FBC  2A A3 03                 LHLD X03A3 } HL STILL POINTS TO INSTRUCTION        } SINCE STRING EVALUATOR USES THESE LOC'S
0FBF  E3                       XTHL
```

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     X0FC1    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 1 CHAR OF ARRAY NAME BACK INTO BC
0FC7  F1              POP   PSW      FETCH DIMENSION CTR INTO ACC FROM STACK
                                     FETCH DIMENSION CTR IN DE, PUT EVALUATED ARG ON 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    X0FC1 }
0FD4  CF              RST   1     }  IF DELIMITER CHAR NOT A COMMA, THEN IT MUST BE A ')'
0FD5  29        DB ')'  DAD H     }  PERFORM SYNTAX CHECK FOR ')'
0FD6  22 05 03        SHLD  X0305    SAVE PTR TO INSTRUCTION IN TEMP STORAGE IN FLAG AREA
0FD9  E1              POP   H     }  FETCH ARRAY TYPE FROM STACK } Ø3A4 IS MODIFIED BY THE
0FDA  22 A3 03        SHLD  X03A3 }  STORE IN FLAG AREA          } STRING EVALUATOR
0FDD  1E 00           MVI   E,H'00' } CLEAR E, CLEARS FLAGS FOR POP PSW (SEE 1012)
0FDF  D5              PUSH  D       } SAVE DIMENSION COUNT ON STACK
                                    (MATCHED BY POP H AT Ø89F)
0FE0  11 E5 F5        LXI   D,XF5E5  DUMMY PUSH H, PUSH PSW
      (ERASE ROM ØF32)               WITH ADDR OF FIRST BYTE IN ARRAY TABLE
0FE3  2A E3 03        LHLD  X03E3    LOAD HL WITH ADDR OF START OF NEXT ARRAY IN TABLE
0FE6  3E 19           MVI   A,H'19'  DUMMY  COMPUTE IN HL ADDR OF BYTE AFTER END OF ARRAY TABLE
                      (DAD D)
0FE8  EB              XCHG        }  LOAD DE WITH ADDR OF BYTE AFTER END OF ARRAY TABLE
0FE9  2A E5 03        LHLD  X03E5 }  HL IS PTR USED TO SEARCH ARRAY TABLE
0FEC  EB              XCHG
0FEC  E7              RST   4        TEST (HL) AGAINST (DE)
0FEE  3A A4 03        LDA   X03A4    LOAD ACC WITH VARIABLE TYPE
0FF1  CA 1F 10        JZ    X101F    IF (HL) = (DE), NO ENTRIES IN TABLE OR ARRAY NOT FOUND IN TABLE
                                     (TABLE EMPTY OR NOT FOUND IN TABLE)
0FF4  BE              CMP   M        TEST VAR TYPE OF ARRAY IN TABLE TO SEE IF MATCH TYPE LOOKING FOR
0FF5  23              INX   H        ADVANCE PTR TO ARRAY TABLE
0FF6  C2 02 10        JNZ   X1002    JMP IF NO MATCH, ARRAY IN TABLE NOT THE ONE WE WANT  (NO MATCH)
0FF9  7E              MOV   A,M      FETCH 2ND CHAR OF ARRAY IN TABLE INTO ACC
0FFA  B9              CMP   C        COMPARE WITH 2ND CHAR OF ARRAY IN INSTRUCTION
0FFB  23              INX   H        ADVANCE PTR TO ARRAY TABLE
0FFC  C2 03 10        JNZ   X1003    JMP IF NO MATCH  (NO MATCH)
0FFF  7E              MOV   A,M      FETCH 1ST CHAR OF ARRAY IN TABLE INTO ACC
1000  B8              CMP   B        COMPARE WITH 1ST CHAR OF ARRAY IN INSTRUCTION
1001  3E 23           MVI   A,A'#'   DUMMY
1002         (INX H)  INX   H        ADVANCE PTR TO ARRAY TABLE
1003  23     X1003    INX   H
1003  23              MOV   E,M   }  IF NO MATCH, DE CONTAINS # OF BYTES TO ADD TO HL PTR
1004  5E              INX   H     }  SKIP OVER THIS ARRAY (NOTICE DAD D AT ØFE7)
1005  23              MOV   D,M   }  OTHERWISE, HL POINTS TO BYTE CONTAINING # OF DIMENSIONS
1006  56              INX   H     }
1007  23              JNZ   X0FE7    LOOP IF ARRAY IN TABLE NOT THE ARRAY TO BE FOUND
1008  C2 E7 0F        LDA   X03A3    TEST Ø3A3, IF HERE, WE ARE PROCESSING A DIM STMT, + WE JUST
100B  3A A3 03        ORA   A     }  FOUND THE ARRAY IN THE TABLE. DIM FORCES Ø3A4 TO AF AT ØEDF
                                     "REDIMENSIONED ARRAY" IF Ø3A3 NOT Ø2ØØ
100E  B7              JNZ   X04B2    PRINT ERROR MESSAGE "REDIMENSIONED ARRAY" IF Ø3A3 NOT Ø2ØØ
                      ('DIM')        DIM FORCED ALL FLAGS TO ØØ AT ØFDD
100F  C2 B2 04        POP   PSW      FETCH DIMENSION COUNT INTO ACC, (ZERO FLAG SET WHEN JZ TO ØFE1 FROM ØF32)
1012  F1              JZ    X08B2    JMP IF ERASE COMMAND
                      ('ERASE')
1013  CA B2 08        SUB   M        TEST IF # OF DIMEN. OF ARRAY IN TABLE MATCHES # OF DIMEN OF ARRAY IN INSTR.
1016  96              JZ    X107A    JMP # OF DIMENSIONS CRITERIA HAS BEEN MET
                      ('LET')
1017  CA 7A 10 X101A  MVI   E,H'09' } PRINT
101A  1E 09           JMP   X04B7   } ERROR MESSAGE "SUBSCRIPT OUT OF RANGE"
101C  C3 B7 04 X101F  MOV   M,A      STORE VAR TYPE AT END OF ARRAY TABLE (NEW ENTRY TO TABLE)
101F  77              INX   H        ADVANCE PTR TO TABLE
[illegible]           [illegible]    [illegible] SET UP DE FOR MULTIPLY LOOP THAT [illegible] HOW MANY BYTES ARE NEEDED [illegible]
```

ADD NEW ARRAY TO ARRAY TABLE

STACK

STACK
HI MEM
1ST ARG
2ND ARG
3RD ARG
⋮
Nth ARG
ARRAY TYPE Ø2 Ø3 Ø4 Ø8
LO MEM

STACK CONTENTS AFTER FALLING OUT OF LOOP AT ØFD1

FORMAT OF ARRAY IN ARRAY TABLE

	LO-MEM	
	VAR. TYPE	= S ARRAY TABLE
	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	
	HI-MEM	

```
1024  F1                    POP   PSW FETCH DIMENSION COUNT INTO ACC, D      FORCED ALL FLAGS TO ØØ AT ØF00
1025  CA EE 08      'ERASE' JZ    X08EE JMP IF ERASE COMMAND (ZERO FLAG SET WHEN JZ TO ØFE1 FROM ØF32)
1028  71                    MOV   M,C  STORE 2 CHAR'S OF ARRAY NAME INTO TABLE, 2ND CHAR, THEN 1ST
1029  23                    INX   H
102A  70                    MOV   M,B
102B  23                    INX   H
102C  4F                    MOV   C,A PUT # OF DIMENSIONS IN ACC.
102D  CD A7 04              CALL  X0487 TEST IF ENOUGH MEMORY FOR (2X # OF DIMENSIONS) AVAILABLE
1030  23                    INX   H  } ADVANCE PTR TO ARRAY TABLE TO POINT TO BYTE WHERE # OF
1031  23                    INX   H  } DIMENSIONS WILL BE STORED
1032  22 CD 03              SHLD  X03CD STORE ADDR OF BYTE CONTAINING # OF DIMENSIONS IN NEW ENTRY
1035  71                    MOV   M,C  STORE BYTE - # OF DIMENSIONS IN ARRAY TABLE
1036  23                    INX   H
1037  3A A3 03              LDA   X03A3  LOAD & TEST BRANCHING FLAG STORED AT Ø3A3
103A  17                    RAL        (Ø3A3) = AF => DIM    (Ø3A3) = ØØ => LET
103B  79                    MOV   A,C PUT # OF DIMENSIONS IN ACC
------------------------------------------------------------------------------------------------------
103C  01 09 00      X103C   LXI   B,X0009 SET DEFAULT SIZE TO 11 DECIMAL IF 'LET' (USING AN ARRAY IN LET
103F  D2 44 10              JNC   X1044 JMP IF PROCESSING A 'LET' INSTRUCTION
1042  C1                    POP   B FETCH AN ARGUMENT FROM STACK (LAST ARG FIRST)
1043  03                    INX   B ADD 1 TO ARG (Øth ELEMENT INCLUDED)
1044  71            X1044   MOV   M,C  STORE 2-BYTES CONTAINING # OF ELEMENTS IN DIMENSION
1045  23                    INX   H                                  BEING PROCESSED
1046  70                    MOV   M,B
1047  23                    INX   H
1048  F5                    PUSH  PSW SAVE DIMENSION (ARGUMENT) CTR ON STACK & FLAGS
1049  CD 32 10              CALL  X1032 MULTIPLY # OF ELEMENTS IN ARRAY (FORM PRODUCT [VAR. TYPE] X ARG1 X
104C  F1                    POP   PSW FETCH DIMENSION CTR FROM STACK + FLAGS
104D  3D                    DCR   A  DECREMENT DIMENSION CTR
104E  C2 3C 10              JNZ   X103C IF NOT ZERO, LOOP TO PROCESS ANOTHER ARGUMENT
------------------------------------------------------------------------------------------------------
1051  F5                    PUSH  PSW SAVE ACC (= ØØ) & CARRY FLAG ON STACK
1052  42                    MOV   B,D  SAVE # OF BYTES TO STORE ARRAY ELEMENTS IN BC
1053  4B                    MOV   C,E
1054  EB                    XCHG  PUT PTR TO 1ST BYTE OF 1ST ELEMENT IN DE, PUT # OF BYTES IN HL
1055  19                    DAD   D  COMPUTE ADDR OF BYTE AFTER END OF ARRAY TABLE WITH NEW ARRAY INSTALLED
1056  DA 1A 10              JC    X101A PRINT ERROR MESSAGE "SUBSCRIPT OUT OF RANGE" IF MULTIPLY PRODUCED
1059  CD 94 04              CALL  X0494 TEST FOR AVAILABLE MEMORY SPACE - ERROR MESSAGE IF NOT ENOUGH
105C  22 E5 03              SHLD  X03E5 STORE ADDR OF BYTE AFTER END OF ARRAY TABLE
105F  2B            X105F   DCX   H       BACK-UP PTR TO ELEMENTS IN ARRAY TABLE
1060  36 00                 MVI   M,H'00' CLEAR ALL BYTES OF EVERY ELEMENT TO Ø
1062  E7                    RST   4  TEST IF (HL) = (DE)
1063  C2 5F 10              JNZ   X105F CONTINUE TO CLEAR ELEMENTS IN ARRAY UNTIL (HL) = (DE)
1066  03                    INX   B ADJUST BC SO THAT # IS # OF BYTES TO STORE ELEMENTS +1 (= # OF BYTES FOR
1067  57                    MOV   D,A CLEAR D REG (ACC IS ØØ WHEN FALLING OUT OF LOOP AT 1Ø4E)
1068  2A CD 03              LHLD  X03CD LOAD HL WITH ADDR OF BYTE CONTAINING # OF DIMENSIONS
106B  5E                    MOV   E,M PUT #-OF-DIMENSIONS BYTE IN E, D IS ØØ
106C  EB                    XCHG  PUT PTR IN DE, # OF DIMENSIONS IN HL
106D  29                    DAD   H MULTIPLY #-OF-DIMENSIONS BY 2
106E  09                    DAD   B ADD #-OF-BYTES TO STORE ALL ELEMENTS IN ARRAY + #-OF-DIMEN. BYTE
106F  EB                    XCHG  PUT PTR BACK IN HL, SUM OF # OF BYTES TOTAL TO STORE ARRAY
1070  2B                    DCX   H  } BACK-UP PTR TO POINT TO "CHAIN ADDR" OF ARRAY
1071  2B                    DCX   H  }
1072  73                    MOV   M,E  STORE 2 BYTES CONTAINING VALUE TO TELL # OF BYTES
1073  23                    INX   H    NEEDED TO STORE: BYTE FOR # OF DIMEN
1074  72                    MOV   M,D                + 2 BYTES FOR EACH DIMEN - # OF ELEMENTS IN EACH DIMEN
1075  23                    INX   H                  + # OF BYTES FOR ALL ELEMENTS
1076  F1                    POP   PSW FETCH ACC (= ØØ) & CARRY FLAG FROM STACK
1077  DA AC 10      'DIM'   JC    X10AC IF CARRY IS SET (see 1Ø37), END OF PROCESSING 1 ARRAY FOR DIM,
------------------------------------------------------------------------------------------------------
107A  47            X107A   MOV   B,A  SET BC TO ØØØØ
107B  4F                    MOV   C,A
107C  7E                    MOV   A,M FETCH BYTE CONTAINING # OF DIMENSIONS INTO ACC
```

```
2700
2701 PRINT "ILLEGAL FUNCTION CALL"
2702
2703
2704
2705
2706 REASON - NEED 2 BYTES TO STORE
2707 EACH (ARG+1) = # OF ELEMENTS IN
2708 THAT DIMENSION
2709
2710
2711                 MULTIPLY LOOP
2712                 CUMULATIVE RESULT IN DE
2713                 MULTIPLIER IN BC
2714                 (DE) = (DE) * (BC)
2715                 2-BYTE INTEGER MULTIPLY
------------------------------------------------------
2716 PRIOR TO A 'DIM' DEFAULTS EACH DIM TO
2717                                  11 DECIMAL
2718
2719
2720
2721
2722
2723
2724
2725 ARG2 X . . . . X ARGN)
2726
2727
2728
------------------------------------------------------
2729
2730
2731
2732
2733
2734 AN OVERFLOW
2735
2736
2737
2738
2739
2740
2741 ELEMENTS + 1 FOR #-OF DIMENSIONS BYTE)
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753 DIMEN
2754
2755
2756 EVENTUALLY IMPLIED JMP TO ØED6
2757
------------------------------------------------------
2758
2759 PROCESSING LET WITH AN
     ARRAY ELEMENT ON LEFT
     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  CA 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  X10A5 JPO  X10A9 JMP IF INTEGER OR DOUBLE-PRECISION               )     2786
10A8  09              DAD  B ADD OFFSET TO 2X OFFSET. ALREADY IN HL (STRING VALUE - 3 BYTES PER ELEMENT)  2787
10A9  C1        X10A9 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
-------------------------------------------------------------------------------------------------
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 01        LHLD X0100 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  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  2815 (NAME OF FCN WITH D7 OF 1ST CHAR SET)  DEF
10DD  CD 83 11        CALL X1183 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
```

```
                              } PUT PTR TO DEFINITION STRING IN   VARIABLE VALUE REGION
10E3  72              MOV   M,D
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
                                  (NAME OF FCN WITH D7 OF 1ST CHAR SET)
10E8  CD C1 11  X10E8 CALL  X11C1 PERFORM SYNTAX CHECK FOR 'FN' TOKEN, FIND OR CREATE A VARIABLE IN THE SYMBOL TABLE
                                  (INTEGER, SNG-, OR DBL-PREC)
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 0FA3)
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 0000, THEN FCN NOT YET DEFINED, PRINT ERROR
                                    MESSAGE "UNDEFINED USER FUNCTION"
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 03CD (IN DEFINITION STRING)
1106  01 CF 2C RST1 DB',' LXI B,X2CCF DUMMY
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 0301 TO 80 TO INDICATE DEF IS USING SYMBOL TABLE SEARCH ROUTINE
1110  32 01 03        STA   X0301   }
1113  CD E0 0E        CALL  X0EE0 FIND VARIABLE-ARG IN THE SYMBOL TABLE (see 0FA3), PUT PTR TO NXT
                                  NON-SPACE CHAR IN DE (SHOULD POINT TO ')' IN DEFINITION STRING
1116  EB              XCHG           PUT PTR TO VARIABLE-ARG VALUE REGION IN HL (see 0CE6)
1117  37              STC            SET CARRY - FORCES RETURN TO 1125 FROM 0CBE ROUTINE (see 0CE6)
1118  C3 BE 0C        JMP   X0CBE MOVE VALUE FROM VARIABLE-ARG VALUE REGION ONTO STACK
                                  STRING TYPE VARIABLE (BRANCH)
111B  D2 8A 1C  X111B JNC   X1C8A IF CARRY NOT SET, MISMATCH IN VARIABLES - (see 0C98 FOR STRING VARIABLE
                                  PRINT ERROR MESSAGE "TYPE MISMATCH"
                                  IF CARRY SET, CAME FROM 1118 - SAVE IDENTITY GROUP OF VARIABLE IN STRING FORMULA TABLE
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) = 0 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 07 11        JNZ   X1107 } REPEAT THE ABOVE PROCEDURE FOR ANOTHER VARIABLE-ARG IN THE
                                    DEFINITION STRING
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 03CD (see 1103)
                                  - IN DEFINITION STRING
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 85 09        CALL  X0985 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
                                  APPROPRIATE VARIABLE VALUE REGIONS
[illegible]
```

```
114E  CF              RST  1  } PERFORM SYNTAX CHECK FOR ')' IN DE. . - STRING            0
114F  29      DB ')'  DAD  H  } THIS GUARANTEES THAT THE # OF ARGS IN EACH STRING MUST MATCH   )
1150  3E 05   PUSH D  MVI  A,H'05' DUMMY                                                  >U
1152  CF              RST  1  } PERFORM SYNTAX CHECK FOR '=' TOKEN IN DEFN STRING          0
1153  38      DB '88' CMP  B  }                                                           8
1154  CD 52 0C        CALL X0C52 EVALUATE DEFN-STRING USING ARGS STUFFED INTO VARIABLE-ARGS OF FCN   (see 1137-1149)
1157  2B              DCX  H  BACK-UP PTR TO DEFN-STRING
1158  D7              RST  2  SCAN TO NEXT NON-SPACE CHAR IN DEFN-STRING - TEST THIS CHAR
1159  C2 A9 04        JNZ  X04A9 IF DELIMITER IN DEFN-STRING IS NOT A NULL OR COLON- PRINT ERROR MESSAGE 'SYNTAX ERROR'
115C  D1              POP  D  PUT PTR TO FCN-STRING TO BE EVALUATED IN DE
115D  F7              RST  6  TEST VALUE-TYPE CODE OF EVALUATED FCN
115F  CA 9F 11        JZ   X119F JMP IF FCN IS STRING TYPE- PUT IDENTITY GROUP FOR FCN VALUE AT END OF STRING FORMULA TABLE - WHETHER OR NOT IT'S ALREADY THERE
1161  F1     *X1161   POP  PSW FETCH (TYPE-CODE -3) FLAGS FROM STACK (see 1126)
1162  CA A3 11        JZ   X11A3 JMP IF VARIABLE VALUE TO BE RESTORED IS STRING TYPE
1165  D2 85 11        JNC  X1185 JMP IF CARRY NOT SET - DONE RESTORING VARIABLE VALUES - POPPED BYTE PUSHED AT 10EF
1168  E1              POP  H  FETCH PTR TO LAST BYTE OF VARIABLE - ARG VALUE REGION FROM STACK (see 1125)
1169  C1              POP  B  FETCH 2 BYTES OF VARIABLE VALUE FROM STACK
116A  70              MOV  M,B } STORE THESE 2 BYTES BACK INTO THE VARIABLE VALUE REGION IN THE
116B  2B              DCX  H   } SYMBOL TABLE
116C  71              MOV  M,C }
116D  FA 61 11        JM   X1161 JMP IF VARIABLE IS INTEGER TYPE - ONLY 2 BYTES TO RESTORE
1170  2B              DCX  H   }
1171  C1              POP  B   } FETCH 2 MORE BYTES FROM THE STACK + STORE THEM IN
1172  70              MOV  M,B } THE VARIABLE VALUE REGION.
1173  2B              DCX  H   }
1174  71              MOV  M,C }
1175  E2 61 11        JPO  X1161 JMP IF VARIABLE IS SNG-PREC TYPE - ONLY 4 BYTES TO RESTORE
1178  2B              DCX  H   }
1179  C1              POP  B   } FETCH 4 MORE BYTES FROM THE STACK + STORE THEM
117A  70              MOV  M,B } IN THE VARIABLE VALUE REGION
117B  2B              DCX  H   }
117C  71              MOV  M,C }
117D  2B              DCX  H   }
117E  C1              POP  B   }
117F  70              MOV  M,B }
1180  2B              DCX  H   }
1181  71              MOV  M,C }
1182  C3 61 11        JMP  X1161 LOOP UNTIL ALL VARIABLES HAVE HAD THEIR VALUES RESTORED
1185  D5     *X1185   PUSH D  PUT PTR TO FCN-STRING TO BE EVALUATED ON STACK .
1186  F5              PUSH PSW PUT TYPE-CODE FOR FCN PRECISION ON STACK (see 10EB-10EF)
1187  F7              RST  6  TEST VALUE TYPE-CODE OF EVALUATED FCN (VALUE IN HOLDING AREA)
1188  11 C8 03        LXI  D,X03C8 LOAD DE WITH ADDR OF LAST 3 LOC'S OF STRING FORMULA TABLE
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)
118E  F1              POP  PSW FETCH TYPE-CODE FOR FCN PRECISION FROM STACK
-----------------------------------------------------------------------------------------
1190  118F  FE E5  DUMMY PUSH H  CPI  H'E5' LET - PUT PTR TO VAR VALUE REGION ON STACK
                                        INT = 2    CINT
                                        CHAR = 3   TEST + RETURN
                                        SNG = 4  - CSGN
                                        DBL = 0    CDBL
      CONVERT VALUE IN HOLDING AREA TO SAME PRECISION AS TYPE-CODE IN ACC
1191  E6 07           ANI  H'07' CLEAR UPPER 5 BITS OF TYPE CODE IN ACC
1193  21 F1 01        LXI  H,X01F1 PUT ADDR 01F1 IN HL                                   !
1196  4F              MOV  C,A    } ADD 2X TYPE CODE TO ADDR IN HL                        O
1197  06 00           MVI  B,H'00' } FETCH ADDR OF SUBR FROM TABLE, JMP TO THAT SUBR
1199  09              DAD  B      }
119A  CD 59 0E        CALL X0E59  }                                                       MY
119D  E1              POP  H  PUT PTR TO VAR VALUE REGION ON STACK
119E  C9              RET     DONE                                                        I
-----------------------------------------------------------------------------------------
119F  21 C8 03 *X119F LXI  H,X03C8 } PUT THIS END OF TABLE ADDR ON STACK TO MOVE FCN-VALUE IDENTITY
11A2  F5              PUSH H       } GROUP AT LAST 3 BYTES OF STRING FORMULA TABLE
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)
11A6  7E              MOV  A,M  PUT CHAR COUNT BYTE OF REMOVED GROUP IN ACC
11A7  22 A8 03        SHLD X03A8 ADJUST FLAG AREA PTR TO NEW VALUE (REMOVES GROUP FROM TABLE)
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 1120 no change of HL in 1222)
11AB  77              MOV  M,A } PUT CHAR COUNT BYTE INTO VARIABLE (1ST BYTE) VALUE REGION
```

INVERSE OF 1106-1128

```
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       X1103 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 'AT' XRA B
11C3  3E 80          MVI  A,H'80' LOAD ACC WITH ₪
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 PTR TO THE FUNCTION DEFINITION

11CD  CD 91 21       CALL X2181 CONVERT ARG VALUE TO A CHAR STRING
1100  CD FC 11       CALL X11FC GENERATE CHAR COUNT BYTE FOR STRING BY COUNTING CHAR'S
1103  CD 7A 13       CALL X137A REMOVE 3 BYTE IDENTITY GROUP FROM FORMULA TABLE, (CONVERTED # NOT IN STRING AREA - REST OF 137A IRRELEVANT)
1106  01 CA 13       LXI  B,X13CA } PUT 13CA ON STACK FOR IMPLIED JMP AT 11ED
1109  C5             PUSH B       }
110A  7E       X11DA MOV  A,M PUT CHAR COUNT BYTE IN ACC
110B  23             INX  H   ADVANCE PTR TO STRING FORMULA TABLE - OR IDENTITY GROUP AT ØED
110C  E5             PUSH H   SAVE ADDR OF PTR TO 1ST CHAR OF CONVERTED-ARG STRING
110D  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
1110  E1             POP  H   PUT ADDR OF PTR TO 1ST CHAR OF CONVERTED-ARG STRING IN HL
1111  4E             MOV  C,M } PUT ADDR OF 1ST CHAR OF CONVERTED-ARG STRING IN BC
1112  23             INX  H   }
1113  46             MOV  B,M }
1114  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)
1117  E5             PUSH H   STORE ADDR Ø3C8 ON STACK
1118  6F             MOV  L,A PUT CHAR COUNT BYTE IN L
1119  CD 6D 13       CALL X136D MOVE CONVERTED-ARG STRING INTO STRING AREA
111C  D1             POP  D   PUT ADDR Ø3C8 IN DE
111D  C9             RET IMPLIED JMP TO 13CA - REMOVE IMPLIED JMP ADDR ØE8C (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  E5       X1200 PUSH H SAVE PTR TO CHAR STRING ON STACK - ADDR OF '"' IN FRONT OF STRING (OR 1 LESS THAN 1ST CHAR)
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
```

TEST IF CURRENT LINE # IS FFFF - IF IT IS, PROGRAM IS NOT RUNNING, AND CURRENT INSTRUCTION IS ILLEGAL IN THE DIRECT MODE

STR$

PUT 3 BYTE GROUP IN STRING FORMULA TABLE - AT END OF TABLE

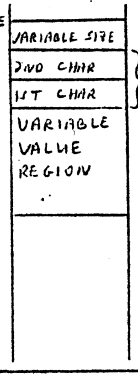
COUNT CHAR'S IN STRING-VALUE IN STRING BEING EVALUATED

PUT CHAR COUNT & PTR TO STRING VALUE IN STRING FORMULA TABLE

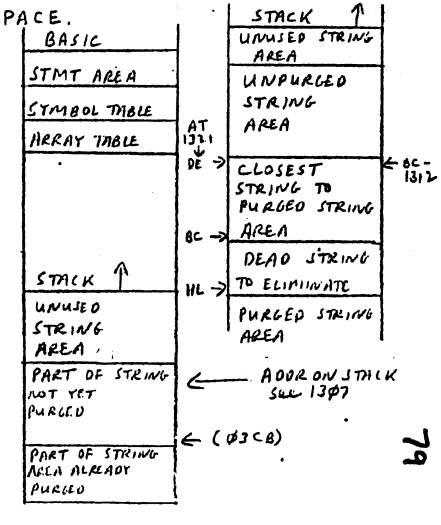
```
                             ENTER AT 11FC OR 11FD
120A  BA                CMP  D       } JMP IF CHAR IS A `"'     } or look for a null if enter at        :
120B  CA 12 12          JZ   X1212   }                          } 11FF from 14E3                         J
120E  BA                CMP  B       } JMP IF CHAR IS NOT `"'                                            8
120F  C2 03 12          JNZ  X1203   }                                                                   D
1212  FE 22    X1212    CPI  A'"'    } IF CHAR IS `"' CALL Ø7D1 - IF TERMINATOR CHAR TO BE FOUND IS A" NULL (AND IT WAS FOUND AT 12ØE) - SCAN TIL NEXT
1214  CC 01 07          CZ   X0701   }                                                       LQ  NON-SPACE CHAR FOUND
1217  E3                XTHL PUT PTR TO END OF STRING ON STACK, PUT PTR TO START OF STRING IN HL (PTR TO `"' - see 12ØØ)
1218  23                INX  H PT TO FIRST CHAR IN STRING                                      #
1219  EB                XCHG  PUT PTR TO 1ST CHAR OF STRING-VALUE IN DE                            - SET-UP FOR ADDITION OF IDENTITY GROUP INTO
121A  79                MOV  A,C PUT CHAR COUNT IN ACC                                               STRING FORMULA TABLE
121B  CD F1 11          CALL X11F1 STORE CHAR COUNT & PTR TO CHAR STRING IN FLAG AREA    M     Ø3C8  Ø3C9  Ø3CA - LAST 3 LOC'S IN
121E  11 CA 03 X121E    LXI  D,X03CA LOAD DE WITH PTR TO FLAG AREA                       H     STRING FORMULA TABLE
1221  3E 05    PUSH D   MVI  A,H'05' DUMMY                                               >U
1223  2A A8 03          LHLD X03A8 } LOAD HL WITH NEXT AVAILABLE BYTE IN STRING FORMULA TABLE     *(
1226  22 12 04          SHLD X0412 } (ADDR OF CHAR COUNT BYTE OF STRING VALUE TO BE ADDED TO STRING FORMULA TABLE)
1229  3E 03             MVI  A,H'03' } SET # OF BYTES TO MOVE IN 1866 TO Ø3           >
122B  32 A4 03          STA  X03A4   } SET VALUE TYPE CODE TO Ø3                      2S
122E  CD 66 18          CALL X1B66  PUT 3 FLAGS IN TABLE Ø3AA - PUT 3 BYTE GROUP IN FORMULA TABLE   M
1231  11 CB 03          LXI  D,X03CB LOAD END-OF-TABLE ADDR INTO DE (1 MORE THAN END OF TABLE)
1234  E7                RST  4     TEST IF HL = DE
1235  22 A8 03          SHLD X03A8 STORE ADDR OF NXT AVAILABLE POSITION IN TABLE Ø3AA       "(
1238  E1                POP  H FETCH PTR TO END OF STRING FROM STACK (RESTORE HL TO PT TO STRING TO BE EVALUATED)
1239  7E                MOV  A,M  PUT NULL IN ACC
123A  C0                RNZ   RETURN IF END OF TABLE NOT YET REACHED                     @
123B  1E 10             MVI  E,H'10' } PRINT ERROR MESSAGE "STRING FORMULA TOO COMPLEX"
123D  C3 B7 04          JMP  X04B7   }                                                   C7
------------------------------------------------------------------------------------------------
1240  23       X1240    INX  H   ADVANCE PTR TO FIRST CHAR OF CONVERTED LINE #             #
1241  CD FC 11 X1241    CALL X11FC GENERATE CHAR COUNT BYTE FOR CHAR STRING SURROUNDED BY QUOTES AND/OR TERMINATED BY A NULL   M
1244  CC 7A 13 X1244    CALL X137A REMOVE THE STRING-TO-PRINT FROM STRING AREA (IF IT'S THERE) - PUT PTR TO CHAR COUNT BYTE IN HL  M
1247  CD 56 19          CALL X1956 PUT 3 BYTE IDENTITY GROUP IN BCD - CHAR COUNT BYTE IN D   MV
124A  14                INR  D  ADVANCE CHAR CTR (TO ENTER LOOP)
124B  15       X124B    DCR  D } DECREMENT CHAR COUNT - EXIT LOOP WHEN Ø               H
124C  C8                RZ     }
124D  0A                LDAX B   FETCH CHAR FROM STRING
124E  CF                RST  3   PRINT CHAR ON OUTPUT CONSOLE                            -
124F  FE 00             CPI  H'00' }                                                     L!
1251  CC A1 0A          CZ   X0AA1 } PRINT NULLS AT END OF LINE (ONLY AFTER CR)
1254  03                INX  B   ADVANCE PTR TO STRING TO PRINT                          CK
1255  C3 4B 12          JMP  X124B LOOP TIL ENTIRE STRING PRINTED                        7
------------------------------------------------------------------------------------------------
1258  B7       X1258    ORA  A  SET FLAGS - CHAR COUNT OF A STRING VALUE IN ACC
1259  0E F1    POP PSW  MVI  C,H'F1' DUMMY
125B  F5                PUSH PSW SAVE FLAGS & CHAR COUNT ON STACK                        *}
125C  2A CD 03          LHLD X03CD LOAD HL WITH ADDR OF START OF STACK (1 BYTE PAST END OF STRING AREA)
125F  EB                XCHG       SAVE END-OF-STRING-AREA ADDR IN DE                    *K
1260  2A CB 03          LHLD X03CB LOAD HL WITH ADDR OF NEXT AVAILABLE BYTE IN STRING SPACE   /
1263  2F                CMA       } FORM 1'S COMPLEMENT OF CHAR COUNT - PUT IN BC        O
1264  4F                MOV  C,A  }
1265  06 FF             MVI  B,H'FF' } - # IS MINUS, THEREFORE B WILL BE FF
1267  09                DAD  B } COMPUTE NEW ADDR FOR NEXT BYTE AVAILABLE
1268  23                INX  H } - COMPLETES 2'S COMPLEMENT OF CHAR COUNT (HL HAS OLD ADDR - # NEW STRING CHAR COUNT)
1269  E7                RST  4   TEST ADDRESS JUST COMPUTED      RC HL<DE
126A  DA 74 12          JC   X1274 IF HL<DE, NEW ADDR < START OF STACK - NOT ENOUGH ROOM - PURGE OLD Z STRINGS FROM STRING AREA TO MAKE ROOM
126D  22 CB 03          SHLD X03CB STORE NEW ADDR FOR NEXT AVAILABLE BYTE IN STRING AREA IN FLAG AREA  "K
1270  23                INX  H ADVANCE PTR TO STRING AREA - POINTS TO 1ST BYTE WHERE NEW STRING WILL BE PUT
1271  EB                XCHG  PUT PTR TO STRING AREA IN DE
1272  F1       X1272    POP  PSW FETCH CHAR COUNT OF STRING FROM STACK                   I
1273  C9                RET
------------------------------------------------------------------------------------------------  FREE-UP STRING SPACE BY ELIMINATING
1274  F1       X1274    POP  PSW PUT CHAR COUNT IN ACC & RESTORE FLAG STATUS FROM STACK       STRINGS WITHOUT AN IDENTITY GROUP
1275  1E 0E             MVI  E,H'0E' } PRINT ERROR MESSAGE IF 2ND TIME THROUGH       J7     IN STRING FORMULA TABLE, SYMBOL TABLE,
1277  CA B7 04          JZ   X04B7   }                                                      OR ARRAY TABLE.
```

```
                                        WHEN IMPLIED JMP TO 125A FROM 130E, 127A WILL PRINT AN
                                        ERROR MESSAGE
127A  BF                    CMP  A   TEST CHAR COUNT, CLEAR CARRY BIT - SET ZERO BIT (  ?     3060
127B  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 1282  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 B6 12               JNZ  X12B6 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               X12B6 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 D0 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  D } 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
12DE  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 4F 12               JZ   X124F JMP IF ALL ELEMENTS OF CURRENT ARRAY HAVE BEEN 'TESTED' BY 12EA  J>  3119
```

SYMBOL TABLE



```
12E6  01 00 12          LXI  B,X1200 FOR ARRAYS, PUT IMPLIED JMP ADDR ON STACK (SIMILAR TO CALL )IZEA 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  B6                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, OR ARRAY TABLES)               M    3131
12F4  2A C8 03               LHLD X03C8 LOAD HL WITH ADDR OF NEXT AVAILABLE BYTE IN STRING AREA (REMEMBER *K   3132 (12BB-12B3) 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 AREA
12FD  E7                                  └ or ADDR OF 1ST CHAR OF STRING CLOSEST TO PURGED STRING      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 STRING IS IN STRING SPACE, BUT (03C8)
1303  C1                     POP  B   PUT RETURN ADDR IN BC                                          A      3145 HAS NOT BEEN UPDATED TO INCLUDE IT
1304  F1                     POP  PSW REMOVE ADDR OF END OF STRING AREA FROM STACK (see 128D) INFO FOR STRING CLOSEST TO PURGED 3146
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 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  78                     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
1312  4E                     MOV  C,M }                                                              N      3160 SPACE.
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                                6      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 78 04               CALL X0478 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
```



```
1320  C3 83 12            JMP  X1283 LOOP UNTIL ENTIRE STRING AREA HAS BEEN PURGED OF 'DEAD' STRINGS   C
132E  C5         X132E    PUSH B STORE PRECEDENCE BYTE OF PREVIOUS OPERATOR ON STACK (see ØC64)   E     STRING VALUE CONCATENATION
132F  E5                  PUSH H  } PUT PTR TO CHAR COUNT OF PREVIOUSLY EVALUATED STRING VALUE         (ØC98)
1330  2A 12 04            LHLD X0412} ON STACK (see 1226) or 8E13 - call this value 1ST ARG   *
1333  E3                  XTHL    } LEAVE PTR TO STRING TO BE EVALUATED IN HL
1334  CD C8 00            CALL X00C8 EVALUATE NEXT PART OF STRING (PART OF STRING AFTER '+') - PUT 3 MK MORE 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 STACK   - VALUE FOUND CALLED 2ND ARG
1338  CC 88 1C            CALL X1C88 TEST THAT 2ND ARG IS A STRING VALUE   H
133B  7E                  MOV  A,M  PUT CHAR COUNT OF 2ND ARG IN ACC
133C  E5                  PUSH H    SAVE ADDR OF CHAR COUNT BYTE OF 1ST ARG ON STACK
133D  2A 12 04            LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT OF 2ND ARG (see 1226)   *
1340  E5                  PUSH H    SAVE ADDR OF CHAR COUNT BYTE OF 2ND ARG ON STACK
1341  8E                  ADD  M    ADD CHAR COUNT OF 2ND ARG TO ACC
1342  1E 0F               MVI  E,H'0F' } IF COMBINED CHAR COUNT PRODUCES AN OVERFLOW
1344  DA 97 04            JC   X0497   } PRINT ERROR MESSAGE "STRING TOO LONG"   27
1347  CD EE 11            CALL X11EE TEST IF ENOUGH ROOM IN STRING AREA FOR CONCATENATED STRING, PUT CHAR COUNT + PTR BYTES IN Ø3C8 Ø3C9 Ø3CA   M
134A  D1                  POP  D    PUT ADDR OF CHAR COUNT BYTE OF 2ND ARG IN DE (DE HAS ADDR - PTS TO FORMULA TABLE)   Q
134B  CD 7E 13  MAYBE NECESSARY  CALL X137E REMOVE 3 BYTE GROUP FROM FORMULA TABLE, CHANGE NXT-BYTE PTR FOR STRING AREA IF 2ND ARG LAST STRING ENTERED IN AREA   N
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 ON STACK
134F  CD 7D 13            CALL X137D REMOVE 3 BYTE GROUP FROM FORMULA TABLE, CHANGE NXT-BYTE PTR FOR STRING AREA IF 1ST ARG LAST STRING IN STRING AREA AT THE MOMENT (see 1391)   M
1352  E5                  PUSH H PUT ADDR OF CHAR COUNT BYTE OF 1ST ARG ON STACK
1353  2A C9 03            LHLD X03C9 } PUT ADDR WHERE CONCATENATED STRING IS TO BE PUT IN DE (ADDR STORED IN   *I
1356  EB                  XCHG       } LAST LOC'S OF FORMULA TABLE IN 11EE AT 1347) (see 11F1 - 11FA)
1357  CD 65 13            CALL X1365 MOVE 1ST-ARG STRING VALUE INTO STRING AREA      STACK BEFORE 1350 - 1361   M      STACK AFTER 1350 - 1361
135A  CD 65 13            CALL X1365 MOVE 2ND-ARG STRING VALUE INTO STRING AREA      [PTR TO STRING]   M               [ØC64]
135D  21 64 0C            LXI  H,X0C64 } PUT ADDR ØC64 ON STACK ABOVE PTR TO     SP -> [PTR TO STRING]   !     [PTR TO STRING] <- SP
1360  E3                  XTHL         } STRING BEING EVALUATED (SEE 1200 or 1222)
1361  E5                  PUSH H       } THIS ADDR IS IMPLIED JMP BACK INTO STRING EVALUATOR AT 123A
1362  C3 1E 12            JMP  X121E PUT 3 BYTE GROUP FOR CONCATENATED STRING IN FORMULA TABLE   C
-----------------------------------------------------------------------------------------------
1365  E1         X1365    POP  H  PUT RETURN ADDR IN HL          MOVE A CHAR STRING INTO STRING
1366  E3                  XTHL PUT RETURN ADDR BACK ON STACK AFTER REMOVING PTR TO STRING FORMULA TABLE IN HL   AREA, DE HAS DEST ADDR PRIOR,
1367  7E                  MOV  A,M FETCH A CHAR COUNT BYTE FROM THE TABLE        REMOVES 1 ADDR FROM STACK
1368  23                  INX  H   } FETCH AN ADDR (POINTS TO 1ST BYTE OF A SELECTED CHAR STRING)   #   ADDR ON STACK SHOULD POINT
1369  4E                  MOV  C,M } FROM TABLE INTO BC   N     TO ADDR CONTAINING A CHAR COUNT
136A  23                  INX  H   }   #     FOLLOWED BY A 2 BYTE PTR ADDR
136B  46                  MOV  B,M }   F     (PTS TO CHAR STRING)
136C  6F                  MOV  L,A PUT CHAR COUNT IN L
136D  2C         X136D    INR  L INCR CHAR COUNT TO ENTER LOOP   .
136E  2D      -->X136E    DCR  L DECR CHAR COUNT   -
136F  C8      <--         RZ     ALL CHAR'S MOVED WHEN CHAR CTR IS Ø - DONE   H
1370  0A                  LDAX B } MOVE 1 BYTE, SOURCE ADDR IN BC, DESTINATION ADDR IN DE
1371  12                  STAX D }
1372  03                  INX  B ADVANCE SOURCE PTR
1373  13                  INX  D ADVANCE DESTINATION PTR
1374  C3 6E 13            JMP  X136E LOOP UNTIL ALL CHAR'S HAVE BEEN MOVED   C
-----------------------------------------------------------------------------------------------
1377  CD 88 1C   X1377    CALL X1C88 PERFORM STRING VALUE CHECK ON VALUE IN HOLDING AREA - PRINT ERROR MESSAGE "TYPE MISMATCH" IF NOT STRING TYPE   H
137A  2A 12 04   X137A    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF LAST GROUP PUT IN FORMULA TABLE   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   BY CHANGING NXT-BYTE PTR
137E  CD 96 13   X137E    CALL X1396 REMOVE A 3 BYTE GROUP FROM STRING FORMULA TABLE (FETCH GROUP, BUT NOT REMOVE, IF DE ≠ ADDR OF CHAR COUNT BYTE OF GROUP IN TABLE   M
1381  EB                  XCHG PUT ADDR OF CHAR COUNT BYTE OF GROUP REMOVED FROM TABLE IN DE
1382  C0      <--         RNZ EXIT HERE IF THE GROUP WAS NOT REMOVED FROM THE FORMULA TABLE (see 139F)   @
1383  D5                  PUSH D PUT ADDR OF CHAR COUNT BYTE OF REMOVED GROUP ON STACK   U
1384  50                  MOV  D,B } PUT PTR ADDR OF GROUP REMOVED FROM TABLE IN DE   P
1385  59                  MOV  E,C }  (POINTS TO 1ST BYTE IN THE RELATED STRING)   Y
1386  1B                  DCX  D BACK-UP PTR TO MAKE IT NXT-BYTE AVAILABLE TYPE OF ADDR (USED IN COMPARISON TO 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 1390)   N    REMEMBER STRINGS ARE UPSIDE DOWN. 1ST CHAR
1388  2A C8 03            LHLD X03C8 LOAD HL WITH NXT-BYTE-AVAILABLE IN STRING SPACE ADDR   *K    IS IN LOWER MEMORY THAN LAST CHAR
138B  E7                  RST  4    TEST IF PTR ADDR'S MATCH
138C  C2 94 13            JNZ  X1394 IF PTR ADDR'S MATCH, GROUP REMOVED SAME AS LAST GROUP PUT IN TABLE   B
138F  47                  MOV  B,A CLEAR B REG FOR DAD   G
```

```
1390  09                 DAD  B COMPUTE NEW NXT-BYTE AVAILABLE ADDR
1391  22 CB 03           SHLD X03CB STORE THIS NEW ADDR IN FLAG AREA (THIS EFFECTIVELY FREES SPACE IN K STRING AREA)
1394  E1         X1394   POP  H PUT ADDR OF CHAR COUNT BYTE OF REMOVED GROUP IN HL
1395  C9                 RET                                                    I
-----------------------------------------------------------------------------------------------
1396  2A A8 03   X1396   LHLD X03A8 FETCH PTR TO CURRENT POSITION IN STRING FORMULA TABLE FROM (FLAG AREA
1399  2B                 DCX  H  BACK UP PTR                                    +    REMOVE A 3 BYTE GROUP FROM
139A  46                 MOV  B,M } FETCH 2 BYTES FROM TABLE INTO BC REG'S      F    STRING FORMULA TABLE
139B  2B                 DCX  H                                                 +
139C  4E                 MOV  C,M } (PUT AN ADDR IN BC, HL POINTS TO CHAR COUNT IN  N   DE CONTAINS AN ADDR, WHICH, IF
                                     TABLE AFTER 139D)
139D  2B                 DCX  H                                                 +
139E  E7                 RST  4  TEST IF HL = DE                                     SAME AS ADDR OF CHAR COUNT BYTE
139F  C0                 RNZ     IF HL ≠ DE RETURN                              @    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  AT 13A3 (see 134A)
13A3  C9                 RET  TABLE PTR (DE HAD TO CONTAIN ADDR OF CHAR COUNT BYTE OF THIS GROUP)  I
-----------------------------------------------------------------------------------------------
13A4  01 04 10           LXI  B,X1004 } PUT ADDR 1004 ON STACK FOR IMPLIED JMP AT 13AF          LEN
13A7  C5                 PUSH B       }                                         E
13A8  CD 77 13   X13A8   CALL X1377 SAME AS FOR LEN ----> IF ARG STRING LAST IN AREA, REMOVE IT, ADDR OF CHAR COUNT BYTE IS IN HL
13AB  AF                 XRA  A } CLEAR D (FOR DAD AT 149B)    CLEAR ACC        /
13AC  57                 MOV  D,A }                                             W
13AD  7E                 MOV  A,M FETCH LENGTH BYTE FROM STRING FCN TABLE | FETCH LENGTH BYTE FROM STRING FORMULA TABLE
13AE  B7                 ORA  A SET FLAGS - TEST FOR NULL STRING                7
                              (used by 13B0-13B3 & 149)
13AF  C9                 RET       | IMPLIED JMP TO ROUTINE TO CONVERT BYTE IN ACC TO 2 BYTE INTEGER IN HOLDING AREA
-----------------------------------------------------------------------------------------------
13B0  CD A8 13           CALL X13A8 IF ARG STRING LAST STRING IN STRING AREA, REMOVE IT, ADDR OF CHAR COUNT BYTE IN HL   ASC
13B3  CA EE 08           JZ   X08EE ZERO FLAG SET-IN 13AE IF NULL STRING - PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL
13B6  23                 INX  H ADVANCE PTR TO STRING FCN TABLE - POINT TO STRING PTR ADDR IN TABLE
13B7  5E                 MOV  E,M } FETCH PTR TO STRING INTO DE FROM STRING FCN TABLE
13B8  23                 INX  H   }
13B9  56                 MOV  D,M }                                             V
13BA  1A                 LDAX D FETCH 1ST CHAR OF STRING INTO ACC
13BB  C3 04 10           JMP  X1004 CONVERT CHAR TO 2 BYTE INTEGER VALUE IN HOLDING AREA   CT
-----------------------------------------------------------------------------------------------
13BE  3E 01              MVI  A,H'01' PUT A CHAR COUNT OF 1 IN ACC (FCN PRODUCES A 1 CHAR STRING)   CHR$
13C0  CD EE 11           CALL X11EE TEST IF ENOUGH ROOM IN STRING AREA FOR 1 CHAR STRING, PUT IDENTITY GROUP AT END OF FORMULA TABLE
13C3  CD 84 14           CALL X1484 CONVERT ARG IN HOLDING AREA TO A 1 BYTE INTEGER (0-255)   *I
13C6  2A C9 03           LHLD X03C9 PUT ADDR WHERE STRING STARTS IN HL (PUT THERE BY 11F1-11F9)
13C9  73                 MOV  M,E  STORE THE 1-BYTE ARG IN THE STRING AREA
13CA  C1         X13CA   POP  B REMOVE IMPLIED JMP ADDR 0E0C FROM STACK (see 0E52-0E55) - NOT NEEDED FOR THIS FCN
13CB  C3 1E 12           JMP  X121E MOVE IDENTITY GROUP INTO STRING FORMULA TABLE, PUT PTR TO STRING TO BE EVALUATED IN HL
-----------------------------------------------------------------------------------------------
13CE  CD 5A 14           CALL X145A PERFORM SYNTAX CHECK FOR ')', MOVE 2ND ARG FROM STACK TO B   LEFT$
13D1  AF                 XRA  A   CLEAR ACC (FLAG FOR LEFT$ FCN) - CHAR'S START AT THE LEFT, OFFSET OF 0 (see 13EE)
13D2  E3         X13D2   XTHL PUT PTR TO CHAR COUNT BYTE IN HL, PUT PTR TO STRING TO BE EVALUATED ON STACK
13D3  4F                 MOV  C,A SAVE OFFSET FOR 1ST CHAR POSITION IN STRING INC
13D4  3E E5     PUSH H   MVI  A,H'E5' DUMMY
13D6  E5         X13D6  ┌PUSH H SAVE PTR TO CHAR COUNT BYTE OF 1ST ARG (STRING) ON STACK
13D7  7E                │MOV  A,M PUT CHAR COUNT BYTE IN ACC (FROM STRING FORMULA TABLE)
13D8  B8                │CMP  B COMPARE CHAR COUNT AGAINST 2ND ARG
13D9  DA DE 13          │JC   X13DE JMP IF 2ND ARG > CHAR COUNT OF 1ST ARG
13DC  78                 MOV  A,B IF 2ND ARG ≤ CHAR COUNT, THEN STRING FROM FCN HAS CHAR COUNT = 2ND ARG - PUT THIS NEW CHAR COUNT IN ACC FOR CALL 1258
13DD  11 0E 00   MVI C,H'0E' LXI D,X000E DUMMY
13E0  C5                ┌PUSH B SAVE BC ON STACK (2ND ARG + BRANCH FLAG)
13E1  CD 58 12          │CALL X1258 TEST IF ENOUGH ROOM IN STRING AREA FOR THE NEW STRING, ADDR OF 1ST CHAR OF NEW STRING (IN STRING AREA) IN DE
13E4  C1                └POP  B RESTORE BC FROM STACK - B IS 2ND ARG C IS OFFSET FOR 1ST CHAR POSITION IN STRING
13E5  E1                 POP  H } PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN HL, LEAVE IT ON THE STACK
13E6  E5                 PUSH H }
13E7  23                 INX  H ADVANCE PTR TO STRING FORMULA TABLE
13E8  46                 MOV  B,M } PUT PTR TO 1ST CHAR OF 1ST ARG STRING IN HL
13E9  23                 INX  H   }
13EA  66                 MOV  H,M }
13EB  68                 MOV  L,B }
13EC  06 00              MVI  B,H'00' CLEAR B FOR DAD INSTR
13EF  09                 DAD  B COMPUTE ADDR OF 1ST CHAR OF STRING AS A RESULT OF THE FCN BEING EXECUTED
```

```
13EF  44           MOV  B,H  } PUT ADDR OF 1ST CHAR OF NEW STRING (RESULT OF FCN) IN BC
13F0  4D           MOV  C,L  }
13F1  CD F1 11     CALL X11F1 PUT 3-BYTE IDENTITY GROUP IN LAST POSITION OF STRING FORMULA TABLE
13F4  6F           MOV  L,A  PUT CHAR COUNT OF NEW STRING IN L
13F5  CD 60 13     CALL X1360 MOVE NEW STRING INTO STRING AREA
13F8  D1           POP  D    PUT PTR TO CHAR COUNT BYTE OF 1ST ARG IN DE
13F9  CD 7E 13     CALL X137E REMOVE THE IDENTITY GROUP FROM FORMULA TABLE IF LAST ONE ENTERED
13FC  C3 1E 12     JMP  X121E PUT 3-BYTE IDENTITY GROUP FOR NEW STRING IN STRING FORMULA TABLE

                                                                                                  RIGHT$
13FF  CD 5A 14     CALL X145A PERFORM SYNTAX CHECK FOR ')', MOVE 2ND ARG FROM STACK TO BC
1402  D1           POP  D  } PUT PTR TO CHAR COUNT BYTE IN DE, LEAVE IT ON STACK
1403  D5           PUSH D  }                OF 1ST ARG
1404  1A           LDAX D  PUT CHAR COUNT OF 1ST ARG (STRING) IN ACC
1405  90           SUB  B  SUBTRACT 2ND ARG - COMPUTE OFFSET FOR 1ST CHAR POSITION OF NEW STRING
1406  C3 D2 13     JMP  X13D2 REST OF THIS FCN SIMILAR TO LEFT$

                                                                                                  MID $
1409  EB           XCHG PUT PTR TO STRING TO BE EVALUATED IN HL (WAS PUT IN DE AT ØE3C)
140A  7E           MOV  A,M PUT DELIMITER CHAR IN STRING TO BE EVALUATED IN ACC (CHAR IS ')' IF ONLY 2 ARGS
140B  CD 5D 14     CALL X145D PUT 2ND ARG IN B, TEST THAT 2ND ARG IS NOT Ø.
140E  C5           PUSH B     PUT 2ND ARG ON STACK
140F  1E FF        MVI  E,H'FF' SET 3RD ARG TO MAXIMUM IF 2ND ARG NOT IN STRING BEING EVALUATED
1411  FE 29        CPI  A')'  TEST DELIMITER CHAR
1413  CA 18 14     JZ   X1418 JMP IF CHAR IS ')' - ONLY 2 ARGS INVOLVED
1416  CF           RST  1  } PERFORM SYNTAX CHECK FOR ',' BETWEEN 2ND & 3RD ARGS
1417  2C           INR  L  }
1418  CD 81 14     CALL X1481 EVALUATE 3RD ARG TO A 1 BYTE INTEGER VALUE IN E
141B  CF           RST  1  } PERFORM SYNTAX CHECK FOR ')'
141C  29           DAD  H  }
141D  F1           POP  PSW PUT 2ND ARG IN ACC
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
141F  01 06 13     LXI  B,X1306 } PUT ADDR 1306 ON STACK FOR IMPLIED JMP AT 1427, 1420, OR 142F
1422  C5           PUSH B       }
1423  3D           DCR  A  DEC 2ND ARG - 1ST CHAR POSITION OFFSET
1424  BE           CMP  M  COMPARE 2ND ARG AGAINST BYTE COUNT OF 1ST ARG STRING
1425  06 00        MVI  B,H'00' FORCE 2ND ARG TO Ø (JUST IN CASE 2ND ARG > CHAR COUNT OF 1ST ARG)
1427  D0    1306 ← RNC  IMPLIED JMP TO 1306 IF 2ND ARG ≥ CHAR COUNT OF 1ST ARG
1428  4F           MOV  C,A  PUT 2ND ARG-1 IN C
1429  7E           MOV  A,M  PUT CHAR COUNT BYTE IN ACC
142A  91           SUB  C  COMPUTE #OF CHAR'S IN ARG STRING STARTING AT 2ND ARG POSITION
142B  BB           CMP  E  COMPARE THIS REMAINDER WITH 3RD ARG
142C  47           MOV  B,A } USE ENTIRE ARG STRING STARTING AT 2ND ARG CHAR POSITION
142D  D8    1306 ← RC       } SINCE 3RD ARG > REMAINDER CHAR COUNT OF ARG STRING
142F  43           MOV  B,E } USE ONLY 3RD-ARG # OF CHAR'S, SINCE 3RD ARG ≤ REMAINDER CHAR
1430  C9    1306 ← RET      } COUNT OF ARG STRING

                                                                                                  INP
1431  CD 84 14     CALL X1484 CONVERT LVALUE IN HOLDING TO 1 BYTE INTEGER IN ACC
1434  32 37 14     STA  X1437 STORE ARGUMENT IN INPUT INSTRUCTION AS PORT #
1437  DB 00        IN   H'00' FETCH A BYTE FROM THE INPUT PORT SPECIFIED BY THE ARGUMENT
1439  C3 04 10     JMP  X1004 CONVERT BYTE IN ACC TO INTEGER FORMAT IN HL - STORE IN HOLDING AREA

                                                                                                  OUT
143C  CD 74 14     CALL X1474 CONVERT 1ST ARG TO PORT #, STORE IN NEXT INSTRUCTION, 2ND ARG
143F  D3 00        OUT  H'00' CONVERTED TO 1 BYTE IN ACC.
1441  C9           RET           → OUTPUT BYTE TO PORT

                                                                                                  WAIT
1442  CD 74 14     CALL X1474 CONVERT 1ST ARG TO PORT #, STORE AT 1453, 2ND ARG CONVERT TO 1 BYTE IN ACC
1445  F5           PUSH PSW 2ND ARGUMENT IS STATUS MASK, SAVE ON STACK
1446  1E 00        MVI  E,H'00' IF NO 3RD ARGUMENT IN INSTRUCTION, DEFAULT 3RD ARG TO ØØ
1448  2B           DCX  H  BACK-UP PTR TO INSTRUCTION
1449  07           RST  2  SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION
144A  CA 53 14     JZ   X1453 JMP IF MULL OR ':', END OF INSTRUCTION, NO 3RD ARG TO PROCESS
144D  CF           RST  1  } SYNTAX CHECK FOR COMMA BETWEEN 2ND & 3RD ARGUMENTS
144E  2C           INR  L  }
144F  CD 81 14     CALL X1481 EVALUATE 3RD ARG TO 1 BYTE VALUE IN ACC
1452  C1           POP  B  PUT MASK BYTE (2ND ARG) IN REG B
1453  DB 00        IN   H'00' INPUT STATUS FROM SELECTED PORT (ARG 1)
```

```
1454 A8          XRA  E  XOR BYTE FROM 3RD ARG (STATUS  BIT ACTIVE LO IF ARG3 ≠ 7Ø)
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 ')'  }
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 ?
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 FIRT 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 ','  }
147F 06 07       MVI  B,H'07' DUMMY RST 2
1481 CD 52 0C X1481 CALL X0C52 EVALUATE ARGUMENT STRING TO # VALUE
1484 CD 60 14 X1484 CALL X1460 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 78          MOV  A,E PUT 1 BYTE ARGUMENT VALUE IN ACC
148D C9          RET
148E CD A8 13    CALL X13A8 IF ARG STRNG 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 = addr
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      THEN
3392 1484 - CONVERT VALUE TO INTEGER
3393      ARGUMENT RANGE Ø - 255
3394      ONLY
3395      ARG IN ACC ON RETURN
3396
3397
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 STACK
3410
3411
3412
3413
3414
3415
3416
3417
3418 of first byte in stmt area, DE = 1st line #
3419
```

VAL

LIST

```
14AD  E1          X14AD  POP  H remove addr of first byte of nxt stmt to list from stack
14AE  D1                 POP  D put line # of last line to list in DE
14AF  4E                 MOV  C,M  } FETCH CHAIN FROM STMT POINTED TO BY HL
14B0  23                 INX  H    } STORE ADDR OF NXT STMT (CHAIN ADDR) IN BC
14B1  46                 MOV  B,M  }
14B2  23                 INX  H  POINT TO LINE # OF STMT
14B3  7A                 MOV  A,D  } TEST FOR DOUBLE NULL (END OF STMT AREA)
14B4  B1                 ORA  C    }
14B5  CA 0E 04           JZ   X040E JMP TO ROUTINE TO PRINT "OK" - RETURN TO BASIC
14B8  CD E8 07           CALL X07E8 TEST FOR CNTL-C HAVING BEEN TYPED, RETURN HERE IF
14BB  C5                 PUSH B SAVE FIRST-BYTE ADDRESS OF NXT STMT
14BC  4E                 MOV  C,M  } FETCH LINE # FROM STMT POINTED TO BY H,L
14BD  23                 INX  H    } STORE IN BC
14BE  46                 MOV  B,M  }
14BF  23                 INX  H POINT TO ASCII AREA OF STMT
14C0  C5                 PUSH B SAVE CURRENT LINE # ON STACK (LINE # of current stmt being
14C1  E3                 XTHL SAVE PTR TO STMT ON STACK, PUT CURRENT LINE # IN HL
14C2  EB                 XCHG AFTER XCHG, DE = CURRENT LINE #          HL = LINE # OF LAST STMT
14C3  E7                 RST  4  RC IF HL < DE (ie present line # not to be listed)
14C4  C1                 POP  B  BC IS PTR TO STMT AREA (ASCII PART OF STMT TO LIST)
14C5  DA 0D 04           JC   X040D NO MORE LINES TO LIST, PRINT "OK" + RETURN TO TELETYPE
14C8  E3                 XTHL PUT LAST LINE # TO LIST BACK ON STACK, PUT FIRST-BYTE ADDRESS OF NXT
14C9  E5                 PUSH H PUT FIRST-BYTE ADDR OF NXT STMT ON STACK
14CA  C5                 PUSH B PUT PTR TO STMT ON STACK
14CB  EB                 XCHG AFTER XCHG, DE = FIRST BYTE ADDRESS OF NXT STMT, HL = LINE # OF
14CC  CD 98 0A           CALL X0A98 PRINT CRLF & NULLS, BEFORE 1ST LINE & AFTER EVERY LINE
14CF  CC 73 21           CALL X2173 CONVERT LINE # TO DECIMAL ASCII FORM & PRINT
14D2  3E 20              MVI  A,A' ' } PRINT SPACE AFTER LINE #
14D4  E1                 POP  H      } HL = PTR TO STMT (ASCII AREA)
14D5  DF                 RST  3
14D6  CD E9 14           CALL X14E9 LOAD LINE BUFR WITH EXPANDED STMT FROM STMT AREA
14D9  21 5A 03           LXI  H,X035A LOAD HL WITH ADDR OF FIRST BYTE OF THE LINE BUFR
14DC  01 AD 14           LXI  B,X14AD } PUT THE RETURN ADDR 14AD ON THE STACK FOR USE
14DF  C5                 PUSH B       } BY A RETURN IN SUBROUTINE 1244
14E0  2B           X14E0 DCX  H DECREMENT PTR TO LINE BUFR
14E1  06 00              MVI  B,H'00' SET TERMINATOR SEARCH CHAR TO BE A NULL
14E3  CD FF 11           CALL X11FF GENERATE A CHAR COUNT BYTE BY COUNTING CHAR'S IN THE STRING TILL
14E6  C3 44 12 implied jmp JMP X1244 PRINT THE CHAR STRING
14E9  01 59 03     X14E9 LXI  B,X0359 LOAD B,C WITH ADDR OF BYTE BEFORE FIRST BYTE OF LINE
14EC  16 E1 POP H        MVI  D,H'E1' DUMMY BYTE TO SKIP POP H TO ENTER LOOP [POP H RESTORES PTR
14EE  7E           X14EE MOV  A,M FETCH A BYTE INTO ACC FROM ASCII AREA OF STMT BEING LISTED
14EF  03                 INX  B ADVANCE PTR TO LINE BUFR; POINT TO NXT BYTE AVAILABLE
14F0  B7                 ORA  A TEST BYTE FOR NULL
14F1  23                 INX  H ADVANCE PTR TO ASCII PART OF STMT
14F2  02                 STAX B STORE CHAR IN LINE BUFR
14F3  C8                 RZ   NULL HAS BEEN FOUND - EXIT
14F4  F2 EE 14           JP   X14EE IF NOT A TOKEN, REPEAT LOOP FOR NXT CHAR IN STMT
14F7  FE 91              CPI  H'91' IS TOKEN BYTE = "ELSE"
14F9  CC 19 1C           CZ   X1C19 → DCX B RET  REMOVES COLON IN FRONT OF ELSE (INSERTED
14FC  D6 7F              SUI  H'7F' CONVERT TOKEN TO OFFSET (# OF RESERVED WRD IN LIST)
14FE  E5                 PUSH H SAVE PTR TO STMT IN STACK
14FF  11 7A 00           LXI  D,X007A LOAD DE WITH ADDR OF FIRST BYTE OF RESERVED WRD TABLE
1502  D5           X1502 PUSH D SAVE ADDR OF FIRST BYTE OF RESERVED WRD BEING SCANNED
1503  F5                 PUSH PSW SAVE RESERVED WRD COUNT IN STACK (ADJUSTED TOKEN)
1504  1A           X1504 LDAX D FETCH A BYTE OF RESERVED WRD
1505  13                 INX  D ADVANCE PTR TO RESERVED WRD
1506  B7                 ORA  A TEST FOR D7 SET (INDICATES BYTE IS LAST CHAR IN WRD)
1507  F2 04 15           JP   X1504 LOOP IF END OF WRD NOT FOUND YET
150A  F1                 POP  PSW } AFTER END OF WRD FOUND, FETCH WRD CTR FROM STACK
150B  3D                 DCR  A   } DECR COUNT BY 1
```

```
3420
3421  Q
3422  N
3423  #
3424  F
3425  #
3426
3427  1
3428  J^  COMMAND MODE
3429  M   NO CHAR TYPED
3430  E
3431  N
3432  #
3433  F
3434  #
3435  E   listed)
3436
3437        TO LIST.
3438
3439  A
3440  Z}  COMMAND MODE
3441        STMT IN HL
3442
3443  E
3444        PRESENT LINE
3445  M
3446  M !
3447  >
3448
3449  -
3450  M
3451  !Z
3452  -
3453  E
3454  +
3455
3456  NULL FOUND
3457  CD
3458  Y   BUFR
3459        TO STMT IN HL]
3460
3461
3462  7
3463  #
3464
3465  H
3466
3467
3468  L8  WHILE CONDENSING INPUT BUFFER)
3469  V
3470
3471
3472  U
3473
3474
3475
3476  7
3477
3478
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 0
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 50 14                JM    X14E0 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
```

```
1518  CD 50 05                CALL  X0550 set argument(s) in stack converting line #'s to 2 byte
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: ADDR OF 1ST BYTE OF LINE AFTER DELETED LINES } HL = FIRST
1529  E5                      PUSH H   }                   ADDR OF 1ST BYTE TO BE DELETED                       } DE = LAST
152A  E7                      RST   4  } TESTS FOR RELATIVE SIZE OF DOUBLE ARGUMENTS (RC IF HL < DE)
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  } 0540 ONTO THE STACK. 0540 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              X1530   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 } ADJUST PTR FOR NXT AVAILABLE BYTE
1547  22 E1 03                SHLD  X03E1 }
154A  C9                      RET
```

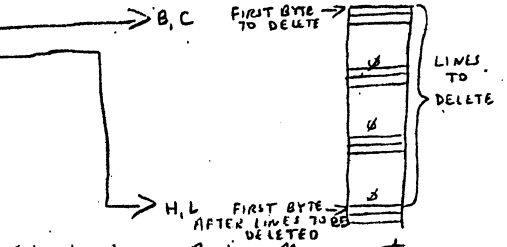
```
154B  CD 11 1C                CALL  X1C11 CONVERT VALUE IN HOLDING AREA TO INTEGER FORMAT, PUT IN HL
154E  7E                      MOV   A,M FETCH SPECIFIED BYTE FROM MEMORY INTO ACC.
154F  C3 04 10                JMP   X1004 CONVERT BYTE IN ACC TO 2-BYTE INTEGER FORMAT STORED IN HOLDING AREA
```

```
1552  CD 6A 14                CALL  X146A 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
```

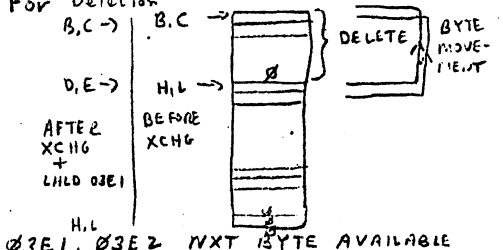
```
155E  CC 53 0C        X155E   CALL  XCC53 EVALUATE STRING VALUE (CHAR'S & VAR'S OK) - 'USING' FORMAT STRING
1561  CC 88 1C                CALL  X1C88 TEST FORMAT VALUE, MUST BE STRING TYPE, OTHERWISE - "TYPE MISMATCH"
1564  CF                      RST   1  } SYNTAX CHECK FOR ';' BETWEEN 'USING' FORMAT STRING & PRINT
1565  3B              DB ';' ~~DCX SP~~ } STMT ARGUMENTS
1566  EB                      XCHG  PUT TO INSTRUCTION STRING IN DE
1567  2A 12 04                LHLD  X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE (IDENTITY GROUP) OF FORMAT STRING
156A  01 01 E8        POP D XCHG LXI  B,XE801 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 00 IF INSTRUCTION IS IMMEDIATE
1570  F5                      PUSH  PSW SAVE RESULT ON STACK
1571  D5                      PUSH  D  SAVE PTR TO INSTRUCTION ON STACK
```

DELETE

2ND OF DOUBLE ARGUMENT



Stmt Area Byte Movement For Deletion



03E1, 03E2 NXT BYTE AVAILABLE IN STMT AREA

PEEK

POKE

FORMATTED OUTPUT ROUTINE

USING

(FOUND BY ISSE - CALL CC53)

LOCATED IN LINE BUFR 035A - 03A1

```
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  3542FUNCTION 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           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  3557MUST 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  3560, 1ST '\' NOT TERMINATED PROPERLY
1594  C1     X1594 POP  H RESTORE HL TO POINT TO CHAR AFTER '\'                           3561
1595  43           MOV  B,E 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 X1598 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 X159F 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 0F 16     JZ   X160F 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
15B0  CA B0 16     JZ   X16B0 IF FORMAT STRING USED-UP, CHECK FOR SIGN CHAR, AND REPEAT USING ENTIRE FORMAT  J-  3578 STRING IF NECESSARY
15B3  FE 2B        CPI  A'+' TEST FORMAT CHAR                                          +  3579
15B5  3E 08        MVI  A,H'08' PUT BIT CODE FOR '+' IN ACC                            >  3580
15B7  CA 9F 15     JZ   X159F JMP TO 159F TO PRINT ANY PREVIOUS '+' CHAR               J  3581
15BA  2B           DCX  H  BACK-UP PTR } FETCH FORMAT CHAR INTO ACC AGAIN              +  3582
15BB  7E           MOV  A,M FETCH CHAR } HL PT TO NXT FORMAT CHAR                          3583
15BC  23           INX  H ADVANCE PTR  }                                               #  3584
15BD  FE 2E        CPI  A'.' TEST FORMAT CHAR                                          .  3585
15BF  CA D0 16     JZ   X16D0 JMP TO 16D0 TO HANDLE '.' AS 1ST FORMAT CHAR FOR A NUMERIC FIELD  J  3586
15C2  FE 5C        CPI  A'\' TEST FORMAT CHAR                                          \  3587
15C4  CA 7F 15     JZ   X157F IF FORMAT CHAR IS '\' - PRINT N-CHAR STRING FIELD OF  JAN  3588ARG STRING VALUE
15C7  0E           CMP  M TEST CURRENT FORMAT CHAR AGAINST NEXT FORMAT CHAR             >  3589
15C8  C2 98 15     JNZ  X1598 JMP IF FORMAT CHAR'S ARE NOT THE SAME                     B  3590↓ NEXT FORMAT CHAR'S MUST BE $$ ** or **$
15CB  FE 24        CPI  A'$' TEST FORMAT CHAR PAIR                                     $  3591
15CD  CA D7 15     JZ   X15E7 JMP IF PAIR IS '$$'                                      J  3592
15D0  FE 2A        CPI  A'*' TEST FORMAT CHAR PAIR                                     *  3593
15D2  C2 98 15     JNZ  X1598 JMP IF PAIR IS NOT '**'                                  B  3594
15D5  78           MOV  A,B PUT FORMAT STRING CHAR COUNT IN ACC                            3595↓ '**'
15D6  FE 02        CPI  H'02' TEST FORMAT STRING CHAR COUNT                                3596
15D8  23           INX  H ADVANCE PTR TO NXT CHAR IN FORMAT STRING                     #  3597
15D9  CA DF 15     JC   X15DF JMP IF LAST CHAR'S IN FORMAT STRING ARE '**' (1ST '*' WAS COUNTED AT 1ST A)  Z  3598
15DC  7E           MOV  A,M PUT NXT FORMAT CHAR IN ACC (CHAR AFTER '**')                   3599
```

```
1500 FE 24        CPI  A'$' TEST FORMAT CHAR
150F 3E 20  X150F MVI  A,A' ' SET 20 BIT IN ACC - BIT FLAG FOR '*' FILL
1511 C2 E8 15     JNZ  X15E8 JMP IF PROCESSING '**' or NOT PROCESSING '**$'
1514 0D           DCR  D DECREMENT FORMAT STRING CHAR COUNT (COUNTS 3RD CHAR - '$')
1515 1C           INR  E ADD 1 TO NUMERIC-FIELD CTR (FOR $)
1516 FE AF        CPI  A'AF'DUMMY          '$$'  '**'  '**$'  - SET 10 BIT IN ACC - BIT FLAG
1518 C6 10  XRA A ADI  A'10' RESULT IN ACC  10   20   30
151A 23           INX  H ADVANCE PTR TO NXT CHAR IN FORMAT STRING (CHAR AFTER '**$' or '$$')
151B 1C     X15E0 INR  E ADD 1 TO NUMERIC-FIELD CTR (1ST $ OF '$$' or 1ST '*' OF '**' or '**$')
151C 82           ADD  D                   '$$'  '**'  '**$'
151D 57           MOV  D,A } RESULT IN D   10+(D) 20+(D) 30+(D)
151E 1C     X15EE INR  E ADD 1 TO NUMERIC-FIELD CTR (2ND CHAR OF '$$', '**', or '**$' or each # FOUND THEREAFTER)
151F 0E 00        MVI  C,H'00' ZERO THE DECIMAL-PT POSITION CTR
1521 05           DCR  B DECREMENT FORMAT STRING CHAR COUNT (COUNTS 2ND CHAR OF '$$', '**', or '**$' or COUNTS FORMAT CHAR
1522 CA 43 16     JZ   X1643 JMP IF FORMAT STRING USED-UP —
1525 7E           MOV  A,M PUT NXT FORMAT CHAR IN ACC
1526 23           INX  H  ADVANCE PTR TO NXT CHAR IN FORMAT STRING
1527 FE 2E        CPI  A'.' TEST FORMAT CHAR
1529 CA 18 16     JZ   X1618 JMP IF FORMAT CHAR IS '.' —
152C FE 23        CPI  A'#' TEST FORMAT CHAR
152E CA EE 15     JZ   X15EE JMP IF FORMAT CHAR IS '#' - ADD 1 COUNT TO NUMERIC-FIELD CTR
1531 FE 2C        CPI  A',' TEST FORMAT CHAR
1533 C2 24 16     JNZ  X1624 JMP IF FORMAT CHAR IS NOT A ',' —
1536 7A           MOV  A,D } SET 40 BIT OF BYTE IN D REG TO INDICATE ',' FOUND ON LEFT SIDE
1537 F6 40        ORI  A'@' } OF '.', IF ANY
1539 57           MOV  D,A
153A C3 EE 15     JMP  X15EE CONTINUE TESTING NEXT FORMAT CHAR
-------------------------------------------------------------------------------------------
1600 7E     X1600 MOV  A,M  PUT NEXT FORMAT CHAR IN ACC
1601 FE 23        CPI  A'#' TEST FORMAT CHAR
1603 3E 2E        MVI  A,A'.' } PUT '.' IN ACC, IF FORMAT CHAR AFTER '.' IS NOT A '#', THEN '.'
1605 C2 98 15     JNZ  X159A } IS NOT A FORMAT CHAR - PRINT '.' & CONTINUE PROCESSING AT START OF NXT FIELD
1608 0E 01        MVI  C,H'01' INITIALIZE DECIMAL-PT POSITION CTR TO 1 (1 PRINT POSITION FOR '.' CHAR)
160A 23           INX  H ADVANCE PTR TO NXT FORMAT CHAR
160B 0C     X1618 INR  C ADD 1 TO DECIMAL-PT POSITION CTR
160C 05           DCR  B DECREMENT FORMAT STRING CHAR COUNT (COUNTS '.' FOUND AT 1600 or 1607 or '#' FOUND AT 161F
160D CA 43 16     JZ   X1643 JMP IF FORMAT STRING USED-UP
1610 7E           MOV  A,M PUT NXT FORMAT CHAR IN ACC
1611 23           INX  H ADVANCE PTR TO NXT CHAR IN FORMAT STRING
1612 FE 23        CPI  A'#' TEST FORMAT CHAR
1614 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 CHAR FOUND
1617 D5     X1624 PUSH D SAVE DE ON STACK
1618 11 41 16     LXI  D,X1641 } PUT ADDR 1641 ON STACK FOR IMPLIED JMP (SAVES MEMORY - 1 BYTE FOR CONDITIONAL RETURNS)
161B D5           PUSH D       }
161C 54           MOV  D,H } SAVE PTR TO FORMAT STRING IN DE (IN CASE 4 '↑' IN A ROW NOT FOUND)
161D 5D           MOV  E,L }
161E FE 5E        CPI  A'↑' TEST FORMAT CHAR
1620 C0           RNZ  IMPLIED JMP TO 1641 IF FORMAT CHAR NOT '↑'
1621 0E           CMP  M TEST 2ND FORMAT CHAR
1622 C0           RNZ  IMPLIED JMP TO 1641 IF 2ND FORMAT CHAR NOT '↑'
1623 23           INX  H ADVANCE PTR TO FORMAT STRING
1624 0E           CMP  M TEST 3RD FORMAT CHAR
1625 C0           RNZ  IMPLIED JMP TO 1641 IF 3RD FORMAT CHAR NOT '↑'
1626 23           INX  H ADVANCE PTR TO FORMAT STRING
1627 0E           CMP  M TEST 4TH FORMAT CHAR
1628 C0           RNZ  IMPLIED JMP TO 1641 IF 4TH FORMAT CHAR NOT '↑'
1629 23           INX  H ADVANCE PTR TO NXT CHAR IN FORMAT STRING (CHAR AFTER '↑↑↑↑'
162A 79           MOV  A,C PUT FORMAT STRING CHAR COUNT IN ACC
162B D6 04        SUI  H'04' SUBTRACT 4 FROM FORMAT STRING CHAR COUNT (FOR '↑↑↑↑')
162D D8           RC   IMPLIED JMP TO 1641 IF CHAR COUNT WAS 3 OR LESS
162E D1           POP  D REMOVE IMPLIED JMP ADDR 1641 FROM STACK
```

```
$  3600
>  3601
8  3602↓ '**$'
   3603
   3604
/  3605
   3606
#  3607
   3608
   3609
W  3610
   3611
   3612
   3613 - COUNTS LAST CHAR OF PREFIX FORMAT GROUP)
JC 3614 AT 15FC('#') or AT 1606 (',')
   3615
#  3616        ##.##
.  3617        **.##
J  3618        **$.##
#  3619        $$.##
J  3620          E  C
,  3621
B$ 3622
   3623↓ ','
@  3624
H  3625
C  3626
   3627↓ '.' - LEADING '.'
#  3628
>. 3629
   3630 FIELD
   3631↓ '.' AS FORMAT CHAR
#  3632
   3633
   3634 AT 161F
JC 3635
   3636
#  3637
#  3638
   3639
U  3640↓
A  3641 FOR CONDITIONAL RETURNS)
U  3642
T  3643
D  3644
^  3645
@  3646
>  3647↓ 1 '↑' FOUND
@  3648
#  3649
>  3650↓ 2ND '↑' FOUND
@  3651
#  3652
>  3653↓ 3RD '↑' FOUND
@  3654
#  3655↓ 4TH '↑' FOUND
   3656
V  3657
X  3658
C  3659
```

```
163C 01                POP  D RESTORE DE FROM STACK (see 1624)   D IS BIT-FLAG BYTE  E IS NUMERIC-FIELD CTR TO LEFT OF DP   D
163D 47                MOV  B,A PUT ADJUSTED FORMAT STRING CHAR COUNT BACK INTO REG B   G
163E 14                INR  D SET Ø1 BIT OF BIT-FLAG BYTE TO INDICATE EXPONENTIAL FORMAT ('↑↑↑↑' FOUND IN FORMAT STRING)
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  X01EB DUMMY RESTORE PTR TO FORMAT STRING TO WHERE IT WAS (see 1629-162A), put it in HL, restore DE from STACK (see 1624)   J O
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)   B
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   TRAILING SIGN CHAR NOT COUNTED
164D B7                ORA  A TEST CHAR COUNT   7
164F CA 63 16          JZ   X1663 JMP IF FORMAT STRING USED-UP   J
1651 7E                MOV  A,M PUT LAST FORMAT CHAR TESTED IN ACC AGAIN
1652 D6 2D             SUI  A'-' TEST FORMAT CHAR   V-
1654 CA 5E 16          JZ   X165E JMP IF FORMAT CHAR IS TRAILING '-'   J^
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 '+'   B
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   F
1660 B2                ADD  D COMBINE THESE NEW BITS WITH FORMER BIT-FLAG BYTE
1661 57                MOV  D,A STORE RESULT IN BIT-FLAG REGISTER   W
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 86 16          JZ   X1686 JMP IF PRINT USING INSTRUCTION IS IMMEDIATE (NOT IN A BASIC PROGRAM)   J6
1668 C5                PUSH B SAVE BC ON STACK   E
1669 D5                PUSH D SAVE DE ON STACK   U
166A CD 52 0C          CALL X0C52 EVALUATE NEXT ARG OF PRINT STMT TO VALUE IN HOLDING AREA   MR
166D D1                POP  D RESTORE DE FROM STACK   Q
166E C1                POP  B RESTORE BC FROM STACK   A
166F C5                PUSH B SAVE REMAINING FORMAT STRING CHAR COUNT (B) + DECIMAL-PT POSITION CTR (C)   EON STACK
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   C
1672 7A                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 ≥ 23, PRINT ERROR MESSAGE 'ILLEGAL FUNCTION CALL' (DBL R-PREC 16 DIGITS + SIGN + DP + EXPONENT = 22)
1679 7A                MOV  A,D PUT BIT-FLAG BYTE IN ACC
167A F6 80             ORI  H'80' SET 80 BIT OF BIT-FLAG BYTE FOR #-CONVERT ROUTINE
167C CD 82 21          CALL X2182 CONVERT ARG-TO-PRINT TO A CHAR STRING   M !
167F CD 41 12          CALL X1241 PRINT CONVERTED-ARG CHAR STRING (COUNTS CHAR'S, REMOVE STRING FROM M ASTRING AREA, PRINTS)
1682 E1        X1682   POP  H PUT PTR TO INSTRUCTION IN HL
1683 2B                DCX  H BACK-UP PTR   +
1684 D7                RST  2 SCAN TO NEXT NON-SPACE CHAR IN INSTRUCTION - RZ IF NULL OR COLON FOUND   H - NON-SPACE CHAR IN ACC
1685 37                STC  SET CARRY - IF END OF INSTRUCTION, FORCES PRINTING CRLF AT 1686   7
1686 CA 94 16          JZ   X1694 JMP IF CHAR IN INSTRUCTION IS A NULL OR COLON   J
1689 FE 3B             CPI  A';' TEST CHAR   ;
168B CA 93 16          JZ   X1693 JMP IF DELIMITER BETWEEN ARGS-TO-PRINT IS A ';' - IGNORE IT   J
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 IGNORE THE DELIMITER   B
1693 D7        X1693   RST  2 SCAN TO NXT NON-SPACE CHAR IN INSTRUCTION   W
1694 C1        X1694   POP  B PUT REMAINING FORMAT STRING CHAR COUNT IN B + DON'T CARE ABOUT C   A
1695 EB                XCHG PUT PTR TO INSTRUCTION IN DE
1696 E1                POP  H } PUT PTR TO IDENTITY GROUP OF FORMAT STRING IN HL (see 1567-1560)
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   U
169A 7E                MOV  A,M PUT ORIGINAL CHAR COUNT BYTE OF FORMAT STRING IN ACC
```

NOTE: LEADING SIGN CHAR IS COUNTED IN VALUE IN E
TRAILING SIGN IS COUNTED IN NEITHER E or C, BUT IDENTIFIED BY Ø4 FLAG BIT

BIT-FLAG BYTE FORMAT
- Ø1 EXPONENTIAL FORMAT '↑↑↑↑'
- Ø2
- Ø4 TRAILING SIGN CHAR '+' or '-'
- Ø8 '+' = 1 '-' = Ø
- 1Ø '$$' 3Ø = '**$'
- 2Ø '**'
- 4Ø COMMA FOUND TO LEFT OF '.'
- 8Ø BIT FLAG FOR # CONVERSION = 1

```
1690 90          SUB  B SUBTRACT REMAINING COUNT OF FORMAT STRING FROM TOTAL CHAR COUNT (OFFSET 3720 TO COMPUTE ADDR OF 1ST REMAINING CHAR)
169C 23          INX  H                                                                  #    3721
169D 4E          MOV  C,M  } PUT ADDR OF 1ST CHAR OF ENTIRE FORMAT STRING IN HL          N    3722
169F 23          INX  H    }                                                             #    3723
169F 66          MOV  H,M  }                                                                  3724
16A0 69          MOV  L,C  }                                                                  3725
16A1 1E 00       MVI  D,H'00' CLEAR D FOR DAD INSTRUCTION                                     3726
16A3 5F          MOV  E,A  PUT DIFFERENCE OF CHAR COUNTS IN E                            -    3727
16A4 19          DAD  D COMPUTE ADDR OF NXT CHAR IN FORMAT STRING                            3728
16A5 78          MOV  A,B  } PUT CHAR COUNT OF REMAINING FORMAT STRING IN B                  3729
16A6 B7          ORA  A    } + TEST IT                                                    7    3730
CONTINUE SAME STRING
16A7 C2 9C 15    JNZ  X159C IF REMAINING STRING NOT USED-UP YET, CONTINUE WITH NXT FIELD OBTAINED FROM 3731 PRESENT POSITION IN STRING
16AA C3 B1 16    JMP  X16B1 SKIP 2 INSTRUCTIONS                                         C1   3732
16AD CD F4 16    X16AD CALL X16F4 PRINT LEADING '+' IF D≠Ø ('+' NOT A FORMAT CHAR IN THIS CASE)  M    3733
16B0 DF          RST  3 PRINT CHAR IN ACC                                               -    3734
16B1 E1          X16B1 POP  H PUT PTR TO INSTRUCTION IN HL                                    3735
16B2 F1          POP  PSW FETCH FLAG STATUS FROM STACK  ↓ IF IMMEDIATE INSTRUCTION FLAG ZERO = 1  3736 - GOES TO 16B6
16B3 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
16B6 DC 98 0A    X16B6 CC  X0A98 IF CARRY-BIT SET (see 1685)                             \    3738 STRING
16B9 E3          XTHL PUT PTR TO IDENTITY GROUP OF FORMAT STRING IN HL, PUT PTR TO INSTRUCTION ON 3739 STACK
16BA CD 7D 13    CALL X137D ERADICATE FORMAT STRING FROM STRING AREA, REMOVE IDENTITY GROUP M FROM 3740 STRING FORMULA TABLE (IF THERE)
16BD E1          POP  H PUT PTR TO INSTRUCTION IN HL                                          3741
16BE C9          RET                                                                    I    3742
-------------------------------------------------------------------------------------------------------------------------------------------
16BF 0E 01       X16BF MVI C,H'01' SET CHAR COUNT FOR LEFT$ (CALL 1305 AT 16DD) TO 1  — STRING 3743 FIELD COUNT = 1
16C1 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
16C3 05          DCR  B DECREMENT FORMAT STRING CHAR COUNT                                    3745                NEEDED SINCE 2ND '\' FOUND)
16C4 CD F4 16    CALL X16F4 PRINT LEADING '+' IF D≠Ø ('+' NOT A FORMAT CHAR IN THIS CASE)  M    3746
16C7 E1          POP  H PUT PTR TO INSTRUCTION IN HL (see 1571)                               3747
16C8 F1          POP  PSW FETCH RESULT-OF-TEST (see 156F) STATUS FROM STACK                   3748
16C9 CA B6 16    JZ   X16B6 JMP IF PRINT USING INSTRUCTION IS IMMEDIATE (NOT IN A BASIC PROGRAM) J6  3749
16CC C5          PUSH B SAVE FORMAT STRING CHAR COUNT (B) & STRING LENGTH (C) ON          ESTACK 3750
16CD CD 52 0C    CALL X0C52 EVALUATE NXT ARG IN PRINT STMT TO VALUE IN HOLDING            MRAREA 3751
16D0 CD 88 1C    CALL X1C88 TEST ARG VALUE - MUST BE STRING TYPE, OTHERWISE "TYPE          M    3752 MISMATCH"
16D3 C1          POP  B  } PUT STRING FIELD COUNT IN C                                    A    3753
16D4 C5          PUSH B  } LEAVE FORMAT STRING CHAR COUNT ON STACK                        E    3754
16D5 F5          PUSH H  PUT PTR TO INSTRUCTION ON STACK                                      3755
16D6 2A 12 04    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE (IDENTITY GROUP) * ON        3756 ARG STRING
16D9 41          MOV  B,C PUT STRING FIELD COUNT (=1) IN B FOR CALL 1305                  A    3757
16DA 0E 00       MVI  C,H'00' PUT 1ST CHAR POSITION OFFSET IN C (LEFT-MOST CHAR           3758 IS 1ST CHAR)
16DC C5          PUSH B SAVE STRING FIELD COUNT FOR LEFT$ ON STACK                        E    3759
16DD CD 05 13    CALL X1305 GENERATE STRING USING LEFT 1-CHAR OF ARG STRING               MU   3760
16E0 CD 44 12    CALL X1244 PRINT THE NEWLY GENERATED STRING                              MD   3761
16E3 2A 12 04    LHLD X0412 LOAD HL WITH ADDR OF CHAR COUNT BYTE OF NEW STRING            *    3762
16E6 F1          POP  PSW PUT STRING FIELD COUNT IN ACC                                       3763
16E7 96          SUB  M   SUBTRACT CHAR COUNT BYTE OF NEW STRING - ALWAYS 2 Ø             3764 RESULT (> Ø IF ARG STRING NOT
16E8 47          MOV  B,A PUT DIFFERENCE COUNT IN B                                       G    3765 LONG ENOUGH TO FILL STRING FIELD)
16E9 3E 20       MVI  A,A' ' PUT A SPACE CHAR IN ACC                                      >    3766
16EB DA 0F 05    RST3 DCR B JC  X050F DUMMY PRINT A SPACE - DECREMENT DIFFERENCE COUNT    Z_   3767
16EF C2 EC 16    JNZ  X16EC LOOP UNTIL SUFFICIENT SPACES HAVE BEEN PRINTED TO FILL ENTIRE 3768 STRING FIELD
16F1 C3 82 16    JMP  X1682 CONTINUE SCAN ARG STRING, IF NOT DONE CONTINUE WITH FORMAT STRING G    3769
-------------------------------------------------------------------------------------------------------------------------------------------
16F4 F5          X16F4 PUSH PSW SAVE ACC ON STACK                                             3770
16F5 7A          MOV  A,D  } PUT BIT-FLAG BYTE IN ACC                                         3771 PRINT LEADING '+' IF D≠Ø
16F6 B7          ORA  A    } + TEST IT                                                    7    3772 '+' NOT A FORMAT CHAR IN THAT
16F7 3E 20       MVI  A,A'+' PUT '+' CHAR IN ACC                                          >+   3773
16F9 C4 18 00    CNZ  X0018 IF BIT-FLAG BYTE ≠ Ø, USE RST 3 TO PRINT '+' CHAR             D    3774 CASE
16FC F1          POP  PSW RESTORE ACC FROM STACK                                              3775
16FD C9          RET                                                                    I    3776
-------------------------------------------------------------------------------------------------------------------------------------------
16FE 32 E9 03    X16FE STA X03E9 CLEAR ERROR BRANCH FLAG (see ØYE8-ØYED)                 2    3777
1701 2A 07 03    LHLD X0307 LOAD HL WITH CURRENT LINE # FROM FLAG AREA                  *W   3778
1704 B4          ORA  H  } TEST IF HL = FFFF                                            4    3779
                 ORA  L  }
```

```
1705  A5          ANA  L  } TEST IF HL = FFFF
1706  3C          INR  A  }
1707  EB          XCHG  PUT CURRENT LINE # IN DE
1708  C8          RZ  RETURN IF LINE # IS FFFF - BASIC IS IN DIRECT COMMAND MODE - EDIT MODE NOT NEEDED
1709  C3 10 17    JMP  X1710 JMP INTO EDIT ROUTINE AUTOMATICLY IF SYNTAX ERROR OCCURRED WHILE RUNNING A BASIC PROGRAM
170C  CC F3 08    CALL X08F3 ASSEMBLE NEXT GROUP OF CHAR INTO A 2 BYTE VALUE IN DE (LINE #)
170F  C0          RNZ  IF NON-SPACE CHAR AFTER LAST DIGIT IS NOT A ':' or NULL IGNORE
1710  E1        X1710  POP  H  REMOVE CNTL-C CHECK ADDRESS FROM STACK
1711  CD 78 05  X1711  CALL X0578 SCAN STMT AREA TO FIND STMT WITH LINE # TO MATCH # IN DE
1714  D2 71 09          JNC  X0971 RNC IF LINE # MATCH NOT FOUND - ERROR MESSAGE - "UNDEFINED STATEMENT"
1717  60                MOV  H,B } PUT ADDR OF FIRST BYTE OF STMT TO BE EDITED IN HL
1718  69                MOV  L,C }
1719  23                INX  H  } ADVANCE PTR TO POINT TO LINE #
171A  23                INX  H  }
171B  4E                MOV  C,M } FETCH LINE # FROM STMT
171C  23                INX  H    } PUT LINE # IN BC
171D  46                MOV  B,M  } ADVANCE PTR TO POINT TO ASCII AREA OF STMT
171E  23                INX  H    }
171F  C5                PUSH B  PUT LINE # IN STACK
1720  CD E9 14          CALL X14E9 MOVE ASCII FROM STMT AREA TO LINE BUFR - EXPAND RESERVED WORDS
1723  E1        X1723  POP  H } PUT LINE # IN HL
1724  F5               PUSH H } LEAVE LINE # ON STACK
1725  CD 73 21         CALL X2173 CONVERT LINE # TO DECIMAL ASCII FORM & PRINT
1728  3E 20            MVI  A,A' ' } PRINT SPACE AFTER LINE #
172A  DF               RST  3       }
172B  21 5A 03         LXI  H,X035A SET HL TO POINT TO FIRST BYTE IN LINE BUFFER
172E  E5               PUSH H  PUT PTR ON STACK
172F  0E FF            MVI  C,H'FF' INITIALIZE CHAR COUNT TO -1
1731  0C        X1731  INR  C   INCR CHAR COUNT (not including null)
1732  7E               MOV  A,M  FETCH CHAR FROM LINE BUFR
1733  E6 7F            ANI  H'7F' STRIP D7 FROM CHAR (MIGHT HAVE COME WITH LAST CHAR OF A RESERVED WORD)
1735  77               MOV  M,A  PUT CHAR BACK IN POSITION IN LINE BUFR
1736  23               INX  H   ADVANCE LINE BUFR POINTER
1737  C2 31 17         JNZ  X1731 LOOP TIL ENTIRE BUFFER HAS BEEN SCANNED
173A  E1               POP  H  RESET HL TO POINT TO FIRST BYTE IN LINE BUFR
173B  47               MOV  B,A CLEAR REG B
173C  16 00       X173C  MVI  D,H'00' CLEAR REG D (DIGIT ACCUMULATOR - ASCII TO BINARY)
173E  CD EC 06    X173E  CALL X06EC  FETCH 1 CHAR FROM INPUT CONSOLE
1741  FE 3A            CPI  A':'  } IF CHAR IS NOT NUMERIC JMP
1743  D2 58 17         JNC  X1758 }
1746  FE 30            CPI  A'0'
1748  DA 58 17         JC   X1758 }
174B  D6 30            SUI  A'0' CONVERT ASCII TO BINARY NUMERIC VALUE
174D  5F               MOV  E,A  SAVE VALUE IN REG E
174E  7A               MOV  A,D  PUT ACCUMULATED VALUE IN ACC
174F  07               RLC  X2  } MULTIPLY VALUE BY 10
1750  07               RLC  X2  }
1751  82               ADD  D+1 }
1752  07               RLC  X2  }
1753  83               ADD  E  ADD NEW VALUE
1754  57               MOV  D,A STORE NEW ACCUMULATED SUM IN D
1755  C3 3E 17         JMP  X173E
1758  E5        X1758  PUSH H      } PUT ADDR 173C ON STACK FOR IMPLIED JMP
1759  21 3C 17         LXI  H,X173C }
175C  E3               XTHL
175D  15               DCR  D  } TEST DIGIT ACCUMULATOR
175E  14               INR  D  }
175F  C2 63 17         JNZ  X1763 } IF DIGIT ACC IS ZERO, NO DIGITS WERE TYPED, SO SET D TO 1
1762  14               INR  D     }
1763  FE 20.    X1763  CPI  A' ' } IF CHAR IS 'SP' JMP
```

EDIT

LINE TO BE EDITED → B,C → DE

AFTER 057B

CHN ADDR
LINE #

Ø
CHN ADDR
LINE #

Ø
CHN ADDR
LINE #

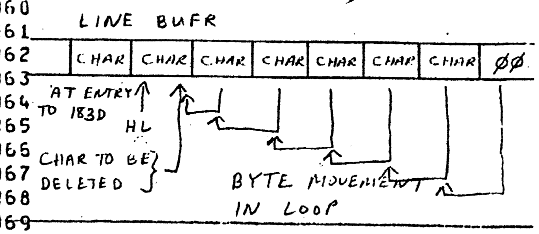
HL - PTR TO LINE BUFR
B - POSITION CTR OF EDITED LINE
C - CHAR CTR OF LINE BUFR (NULL NOT COUNTED)
D - COMMAND CTR (REPEAT N TIMES)
E - SEARCH CHAR STORAGE

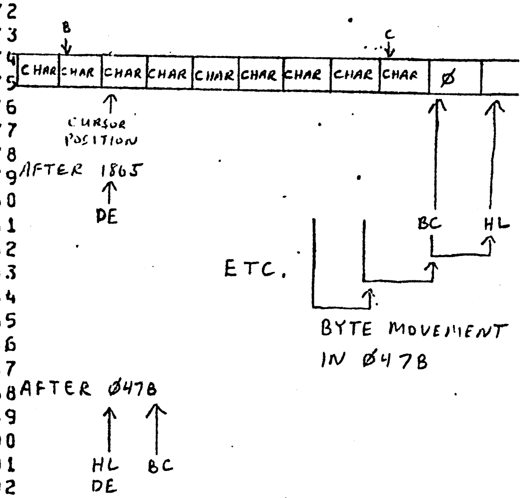
1 BYTE ASCII DIGIT STRING TO BINARY CONVERSION

```
1765  CA AF 17          JZ    X17AF} IF CHAR IS '␣'         . SKIP OVER A CHAR                J/
1768  FE 51             CPI   A'Q' } IF CHAR IS 'Q'        RESTORE LINE TO ORIGINAL          Q   FORM, IGNORE EDIT CHANGES
176A  CA F0 07          JZ    X07F0}                                                          J
176D  FE 4C             CPI   A'L' } IF CHAR IS 'L'        TYPE REMAINDER OF LINE            L   + RE-ENTER EDIT MODE
176F  CA E9 17          JZ    X17E9}                                                          J
1772  FE 53             CPI   A'S' } IF CHAR IS 'S'        SEARCH                            S
1774  CA C0 17          JZ    X17C0}                                                          J@
1777  FE 49             CPI   A'I' } IF CHAR IS 'I'        INSERT CHAR'S                     I
1779  CA 21 18          JZ    X1821}                                                          J!
177C  FE 44             CPI   A'D' } IF CHAR IS 'D'        DELETE CHAR'S                     D
177E  CA F3 17          JZ    X17F3}                                                          J
1781  FE 00             CPI   H'00'} IF CHAR IS CR         END OF EDIT, TYPE REST OF LINE
1783  CA 70 18          JZ    X1870}                                                          J
1786  FE 43             CPI   A'C' } IF CHAR IS 'C'        CHANGE CHAR'S IN LINE             C
1788  CA 0A 18          JZ    X180A}                                                          J
178B  FE 45             CPI   A'E' } IF CHAR IS 'E'        END OF EDIT, DO NOT TYPE          E   REST OF LINE
178D  CA 80 18          JZ    X1880}                                                          J
1790  FE 58             CPI   A'X' } IF CHAR IS 'X'        INSERT AT END OF LINE             X
1792  CA 1C 18          JZ    X181C}                                                          J
1795  FE 4B             CPI   A'K' } IF CHAR IS 'K'        SEARCH, DELETE CHAR'S             K   PASSED OVER WHILE SEARCHING
1797  CA BA 17          JZ    X17BA}                                                          J:
179A  FE 48             CPI   A'H' } IF CHAR IS 'H'        INSERT, DELETE REST OF LINE       H   NOT YET TYPED
179C  CA 19 18          JZ    X1819}                                                          J
179F  FE 7F             CPI   H'7F'} IF CHAR IS RUBOUT     DELETE A CHAR (DURING INSERT)
17A1  CA 71 18          JZ    X1871}                                                          J   RE-ENTER EDIT MODE
17A4  FE 41             CPI   A'A' } IF CHAR IS 'A' DO NOT RETURN  RESTORE ORIGINAL LINE +   A   CHAR FROM INPUT CONSOLE
17A6  C0                RNZ   IF NOT 1 OF ABOVE COMMANDS, IGNORE CHAR TYPED. FETCH ANOTHER       FOR EDIT COMMAND
-----------------------------------------------------------------------------------------------------------
17A7  C1                POP   B  REMOVE 173C ADDR FROM STACK  THIS REMOVES UNWANTED ENTRY A FROM STACK   'A' COMMAND - RESTORE ORIGINAL LINE +
17A8  D1                POP   D  REMOVE LINE # FROM STACK - THIS IS USED IN Ø57B         C                RE-ENTER EDIT MODE
17A9  CD 98 0A          CALL  X0A98 PRINT CRLF + NULLS                                   M
17AC  C3 11 17          JMP   X1711 - LOOP BACK + START EDIT AGAIN                       C
-----------------------------------------------------------------------------------------------------------
17AF  7E         ->X17AF MOV  A,H FETCH CHAR FROM LINE BUFR                                  '␣' COMMAND - SKIP OVER A CHAR
17B0  B7                ORA   A    TEST CHAR                                             7
17B1  C8                RZ         IF NULL, END OF LINE  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND   H
17B2  04                INR   B    INCREMENT CHAR CTR
17B3  DF                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   B/
17B9  C9                RET        IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR   I
-----------------------------------------------------------------------------------------------------------
17BA  E5          X17BA PUSH  H       } PUT 18Ø6 ON STACK FOR IMPLIED JMP     STACK          'K' COMMAND - SEARCH, DELETE CHAR'S
17BB  21 06 18          LXI   H,X1806 } HL STILL PTS TO LINE BUFR             173C   !        PASSED OVER WHILE SEARCHING
17BE  E3                XTHL          }                                       18Ø6
17BF  37                STC      . SET CARRY TO INDICATE 'K' COMMAND WHILE PASSING THRU 'S' COMMAND   7
-----------------------------------------------------------------------------------------------------------
17C0  F5          X17C0 PUSH  PSW  SAVE EDIT COMMAND CHAR ON STACK                           'S' COMMAND - SEARCH
17C1  CD EC 06          CALL  X06EC FETCH SEARCH CHAR FROM INPUT CONSOLE                 M
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        5
17C7  34                INR   M  }                                                       4
17C8  C8                RZ       IF CHAR IS NULL, END OF LINE, Nth SEARCH CHAR NOT FOUND YET   H
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  DF         ->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 D8 17          JNC   X17D8 CARRY NOT SET IF 'S' COMMAND                         RX
17D4  CC 3D 18          CALL  X183D DELETE CHAR POINTED TO BY HL, ADJUST LINE BUFR + CHAR CTR (REG C)   M=
17D7  C2 23 04  INX H INR B JNZ X0423 DUMMY FOR 'K' COMMAND  ADVANCE LINE BUFFER PTR, INCREMENT EDITED LINE CHAR CTR FOR 'S' COMMAND   B@
```

```
170A  7E                  MOV  A,M FETCH CHAR FROM LINE BUFFER ('S COMMAND - NXT CHAR: 'K'-NXT CHAR   BECAUSE BUFR WAS ADJUSTED DOWNWARD
170B  B7                  ORA  A   TEST CHAR                                                7
170C  CA E7 17            JZ   X17E7 JMP IF CHAR IS NULL - END OF LINE, NTH CHAR NOT FOUND  J   'K' COMMAND WILL PRINT '/'
170F  BB                  CMP  E   COMPARE CHAR FROM LINE BUFFER WITH SEARCH CHAR           ;   THEN JMP TO 173C
17E0  C2 CE 17            JNZ  X17CE IF NO MATCH, PRINT CHAR + TRY NEXT CHAR                BN
17E3  15                  DCR  D   DECREMENT COMMAND CTR
17E4  C2 CE 17            JNZ  X17CE IF NOT ZERO, NTH OCCURENCE OF SEARCH CHAR NOT FOUND    BNYET
17E7  F1         X17E7    POP  PSW REMOVE EDIT COMMAND CHAR FROM STACK (SEARCH IS FINISHED OR END OF LINE REACHED)
17E8  C9                  RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR - CURSOR PTS TO NTH SEARCH CHAR OR NULL   I
17E9  CD E0 14   X17E9    CALL X14E0 PRINT REST OF LINE STARTING WITH CHAR POINTED TO BY HL   H   'L' COMMAND - TYPE REMAINDER OF LINE +
17EC  CD 98 0A            CALL X0A98 PRINT CR LF + NULLS                                     M       RE-ENTER EDIT MODE
17EF  C1                  POP  B   REMOVE 173C ADDR FROM STACK                               A
17F0  C3 23 17            JMP  X1723 START EDIT ON NEW LINE - TYPE LINE #, ETC.             C@
17F3  7E         X17F3    MOV  A,M  FETCH CHAR FROM LINE BUFR                                    'D' COMMAND - DELETE CHAR'S
17F4  B7                  ORA  A    TEST CHAR                                                7
17F5  C8                  RZ        RETURN IF CHAR IS NULL (CANNOT DELETE N CHAR'S IF NXT IS NULL)   HCHAR
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                                                7
17FB  CA 06 18            JZ   X1806 IF CHAR IS A NULL, COMMAND IS FINISHED                 J
17FE  DF                  RST  3    PRINT CHAR ON OUTPUT CONSOLE                             -
17FF  CD 30 18            CALL X1830 DELETE CHAR POINTED TO BY HL, ADJUST LINE BUFR + CHAR CTR (REG=C)   M
1802  15                  DCR  D    DECREMENT COMMAND CTR
1803  C2 F9 17            JNZ  X17F9 IF COMMAND CTR IS NOT Ø, REPEAT FOR NXT CHAR           B
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        I
180A  7E         X180A    MOV  A,M  FETCH CHAR FROM LINE BUFR                                    'C' COMMAND - CHANGE CHAR'S IN LINE
180B  B7                  ORA  A    TEST CHAR                                                7
180C  C8                  RZ        JMP TO 173C IF NULL                                      H   CHANGE A CHAR AT CURSOR POSITION
180D  CC EC 06            CALL X06EC FETCH CHAR TO PUT IN LINE BUFR AT CURSOR POSITION      M   TO A CHAR TYPED AFTER 'C'
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                              B
1818  C9                  RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR        I
1819  36 00      X1819    MVI  M,H'00' PUT A NULL AT CURSOR POSITION - DELETES REST OF LINE 6   'H' COMMAND - INSERT MODE, DELETE REST OF
181B  48                  MOV  C,B PUT EDITED-LINE CHAR CTR IN LINE BUFR CHAR CTR           H       LINE NOT YET TYPED
181C  16 FF      X181C    MVI  D,H'FF' SET COMMAND CTR TO FF (MAXIMUM #)                         'X' COMMAND - INSERT MODE AT END OF LINE
181E  CD AF 17            CALL X17AF SKIP OVER A CHAR ('␣' COMMAND) FF TIMES; ENTER INSERT   M/
1821  CD EC 06   X1821    CALL X06EC  FETCH A CHAR FROM INPUT CONSOLE                       H   'I' COMMAND - INSERT CHAR'S
1824  FE 0D               CPI  H'0D'  } IF CHAR IS CR - END OF EDIT; TYPE REST OF LINE
1826  CA 70 18            JZ   X1870  }                                                      J
1829  FE 1B               CPI  H'1B'  } IF CHAR IS 'ESCAPE' - IMPLIED JMP TO 173C, FETCH ANOTHER EDIT COMMAND CHAR
182B  C8                  RZ          }                                                      H
182C  FE 5F               CPI  A'_'   } JMP IF CHAR IS NOT A '←'   } IF CHAR IS A '←', DELETE CHAR
182E  C2 4C 18            JNZ  X184C  }                            } TO LEFT OF CURSOR      BL
1831  05                  DCR  B      } TEST EDITED LINE CHAR CTR  }
1832  04                  INR  B      }
1833  CA 54 18            JZ   X1854 IF CTR IS ZERO, NO CHAR'S TO LEFT OF CURSOR TO DELETE  ↑T
1836  DF                  RST  3    ECHO '←' TO INDICATE A DELETED CHAR      RING BELL      -
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 'CR' OR 'ESCAPE')  U
183D  E5         X183D    PUSH H  SAVE CURSOR POSITION (PRESENT POSITION IN LINE BUFR)
183E  0D                  DCR  C  DECREMENT CHAR COUNT BY 1
```

```
183F 7E        →X183F  MOV  A,M FETCH CHAR FROM LINE BUFFER
1840 B7                ORA  A   TEST CHAR                                          7
1841 CA 1C 1A          JZ   X1A1C=POP H&RET - RESTORE PTR TO LINE BUFR             J
1844 23                INX  H  } FETCH NEXT CHAR FROM LINE BUFR                    #
1845 7E                MOV  A,M}                                                   
1846 2B                DCX  H   BACK-UP PTR. 1 POSITION                            +
1847 77                MOV  M,A STORE CHAR IN NEW POSITION IN BUFR                 
1848 23                INX  H   ADVANCE PTR                                        #
1849 C3 3F 18          JMP  X183F LOOP TIL NULL FOUND                              C?
184C F5        →X184C  PUSH PSW SAVE CHAR TO INSERT ON STACK
184D 79                MOV  A,C } TEST CHAR CTR OF LINE BUFR
184E FE 48             CPI  A'H'} MUST BE LESS THAN 72 FOR NEW CHAR TO BE INSERTED  H
1850 DA 5A 18          JC   X185A JMP IF ENOUGH ROOM IN LINE BUFR FOR AN ADDITIONAL CHAR  ZZ
1853 F1                POP  PSW REMOVE CHAR FROM STACK - RING BELL + IGNORE FURTHER INSERTED CHARS
1854 3E 07     →X1854  MVI  A,H'07'} PRINT CNTL-G ON OUTPUT CONSOLE (RING BELL)     >
1856 DF                RST  3      }                                               
1857 C3 21 18          JMP  X1821 JMP BACK TO START OF INSERT COMMAND              C!
185A 90        →X185A  SUB  B SUBTRACT CURSOR POSITION COUNT FROM LINE LENGTH
185B 0C                INR  C INCREMENT CHAR CTR OF LINE BUFR
185C 04                INR  B INCREMENT EDITED-LINE CHAR CTR
185D C5                PUSH B SAVE BOTH CTR'S ON STACK                             E
185E EB                XCHG  PUT PTR TO LINE BUFR (CURSOR POSITION) IN DE
185F 6F                MOV  L,A  } COMPUTE ADDR OF LAST CHAR IN BUFR (ADDR OF NULL)
1860 26 00             MVI  H,H'00'}                                               &
1862 19                DAD  D
1863 44                MOV  B,H  } PUT ADDR OF LAST CHHR IN BUFR IN BC             D
1864 4D                MOV  C,L  }                                                 M
1865 23                INX  H   ADVANCE PTR TO POINT TO BYTE AFTER NULL IN LINE BUFR  #
1866 CD 78 04          CALL X0478 MOVE CHAR IN LINE BUFR TO MAKE ROOM FOR NEW CHAR  M
1869 C1                POP  B  RESTORE CTR VALUES TO B+C                           A
186A F1                POP  PSW PUT CHAR TO INSERT BACK INTO ACC
186B DF                RST  3 ECHO CHAR ON OUTPUT CONSOLE                          _
186C 77                MOV  M,A STORE CHAR AT CURSOR POSITION IN LINE BUFR
186D 23                INX  H ADVANCE CURSOR POSITION                              #
186E C3 21 18          JMP  X1821 LOOP WITHIN INSERT UNTIL 'J' OR 'ESCAPE'         C!
1871 78        →X1871  MOV  A,B} TEST EDITED-LINE CHAR COUNT            'RUBOUT' COMMAND - DELETE A CHAR DURING INSERT
1872 B7                ORA  A  }                                                   7
1873 C8                RZ    IF ZERO - CANNOT DELETE ANY CHAR'S TIL CURSOR IS MOVED  H
1874 05                DCR  B DECREMENT EDITED LINE CHAR COUNT
1875 2B                DCX  H BACK-UP PTR 1 CHAR                                   +
1876 7E                MOV  A,M} FETCH CHAR FROM BUFR
1877 DF                RST  3  } ECHO CHAR ON OUTPUT CONSOLE                       _
1878 15                DCR  D  DECREMENT COMMAND CTR
1879 C2 71 18          JNZ  X1871 IF NOT ZERO, REPEAT FOR ANOTHER CHAR             B
187C C9                RET  IMPLIED JMP TO 173C - FETCH ANOTHER EDIT COMMAND CHAR  I
1870 CD E0 14   X1870  CALL X14E0 PRINT REST OF LINE STARTING WITH CHAR POINTED TO BY HL  M  'CR' COMMAND- END OF EDIT, TYPE REST OF LINE
1880 CD 98 0A   X1880  CALL X0A98 PRINT CRLF + NULLS                               M  '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)  A
1884 D1                POP  D PUT LINE # OF EDITED STMT IN DE (FOR ROUTINE TO STORE EDITED LINE)  Q
1885 37                STC   } FORCE CARRY SET + PUT CARRY STATUS ON STACK         7
1886 F5                PUSH PSW}(FOOLS COMMAND MODE MODE PROCESSOR, STRING IN LINE BUFR NOT TO BE TREATED AS A DIRECT COMMAND)
1887 21 5A 03          LXI  H,X035A LOAD HL WITH ADDR OF START OF LINE BUFR        !Z
1888 C3 D3 0?          JMP  X0?D3 JMP TO COMMAND MODE PROCESSING ROUTINE (MOVE EDITED LINE FROM LINE BUFR TO STMT AREA - SAME AS NEW LINE FROM KYBRD)
1889 21 45 29   X1880  LXI  H,X2945 LOAD HL WITH ADDR OF SNG-PREC CONSTANT = .5     !E%  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)  MT
1893 C3 9C 18          JMP  X189C ADD CONSTANT = .5 TO # IN HOLDING AREA (or other constant)  C
1896 CD 54 19   X1896  CALL X1954 PUT SNG-PREC CONSTANT IN BCDE  (FETCH FROM MEMORY)  MT
1899 CD 13 1?   X1899  CALL X1?13 NEGATE FLOATING-PT NUMBER IN HOLDING AREA        M   SNG-PREC FLOATING-PT SUBTRACT
189C 78         X189C  MOV  A,B { TEST EXPONENT BYTE OF # IN BCDE                  7   SNG-PREC FLOATING-PT ADDITION
                                                    (# IN BCDE) + (# IN HOLDING ...) ADD
                                                    (# IN BCDE) - (# IN HOLDING ...) SUB
```





```
189E  C8                    RZ     ADD OR SUB DONE IF # IS Ø, RESULT ALREADY IN HOLDING AREA      H
                                   (DONE - 1 # 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 18              CALL   X1836 PUT FLOATING-PT # IN HOLDING AREA ON STACK               M6
18B0  EB                    XCHG   RESTORE CONTENTS OF DE FROM HL
18B1  CD 46 18              CALL   X1846 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:
                                   (DONE - # TOO SMALL)    23 BITS OF VALUE + 1 BIT IMPLIED = 24)
18B9  F5                    PUSH   PSW SAVE EXPONENT DIFFERENCE ON STACK
18BA  CD 73 18              CALL   X1873 COMPUTE XNOR OF SIGN BITS IN ACC, RESTORE LEADING 1 IN MANTISSA'S OF BOTH #'S   H
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 S_M ⊕ 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
18CF  23                    INX    H ADVANCE HL TO POINT TO EXPONENT BYTE Ø415                     #
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>
                                   (ALMOST DONE - 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
18D9  C3 22 19              JMP    X1922 PERFORM ROUNDOFF + NORMALIZE RESULT                       C"
------------------------------------------------------------------------------------------------
18DC  AF          X18DC     XRA    A CLEAR ACC                                                     /
18DD  90                    SUB    B  }  COMPUTE DIFFERENCE OF THE 2 MANTISSA'S
18DE  47                    MOV    B,A}                                                            G
18DF  7E                    MOV    A,M}     (Ø414)   (Ø413)   (Ø412)   ØØ   <- LOST-BIT BYTE
18E0  9B                    SBB    E  }     - C      - D      - E      - B
18E1  5F                    MOV    E,A}     ---      ---      ---      ---
18E2  23                    INX    H  }      C        D        E        B                          #
18E3  7E                    MOV    A,M}
18E4  9A                    SBB    D  }
18E5  57                    MOV    D,A}                                                            W
18E6  23                    INX    H  }                                                            #
18E7  7E                    MOV    A,M}
18E8  99                    SBB    C  }
18E9  4F                    MOV    C,A}                                                            O
18EA  DC 4F 19    X18EA     CC     X194F IF CARRY SET (OVERFLOW) (OR MINUS #), CHANGE SIGN OF RESULT (STORED AT Ø416 BY X1873)
                                   + COMPUTE Ø - (CDEB)
------------------------------------------------------------------------------------------------
18ED  68          X18ED     MOV    L,B } CHANGE REGISTERS HOLDING # FROM CDEB TO CDHL (SPEEDS UP BIT SHIFT
18EE  63                    MOV    H,E }                                                  ROUTINE)
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
18F6  4A                    MOV    C,D }  # IN C D H L (ACC ØØ)                                    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)  4086
1908  29                 DAD   H DO A QUICK LEFT SHIFT (1 BIT LEFT) ON THE 2 LO BYTES IN HL (# IS IN CDHL)  4087
1909  7A                 MOV   A,D  } SHIFT BYTE IN D LEFT 1 BIT                                  4088
190A  17                 RAL        }                                                             4089
190B  57                 MOV   D,A  }                                                             4090
190C  79                 MOV   A,C  } SHIFT BYTE IN C LEFT 1 BIT                                  4091
190D  8F                 ADC   A    }                                                             4092
190E  4F                 MOV   C,A  }                                                             4093
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  4094
1912  78                 MOV   A,B PUT SHIFT CTR IN ACC FROM B                                    4095
1913  5C                 MOV   E,H  } CHANGE REGISTERS HOLDING # FROM CDHL TO CDEB (STANDARD FORMAT)  4096
1914  45                 MOV   B,L  }                                                             4097
1915  B7                 ORA   A TEST SHIFT CTR                                                   4098
1916  CA 22 19           JZ    X1922 IF SHIFT CTR IS ZERO, PERFORM ROUNDOFF, STORE RESULT IN HOLDING AREA  4099
1919  21 15 04           LXI   H,X0415 LOAD HL WITH ADDR OF EXPONENT BYTE                          4100
191C  86                 ADD   M  } COMPUTE EXPONENT = EXPONENT - (# OF SHIFTS TO NORMALIZE).     4101
191D  77                 MOV   M,A}                                                               4102
191E  D2 02 19           JNC   X1902 IF NO CARRY, NORMALIZED # IS SO SMALL THAT IT IS SAME AS Ø   4103
1921  C8            DONE RZ    IF EXPONENT BYTE IS ZERO, NORMALIZED # SAME AS Ø, ADD IS DONE      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 F1B46)  4109
192B  23                 INX   H ADVANCE PTR TO LOC Ø416 (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)         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 87 04           JMP   X0487   }                                                          4126
1943  7E          X1943  MOV   A,M  } ADD LO BYTE OF # TO E                                       4127
1944  83                 ADD   E    }                                                             4128
1945  5F                 MOV   E,A  }                                                             4129
1946  23                 INX   H ADVANCE PTR TO #                                                 4130
1947  7E                 MOV   A,M  }                                                             4131
1948  8A                 ADC   D    } ADD MED BYTE OF # TO D                                      4132
1949  57                 MOV   D,A  }                                                             4133
194A  23                 INX   H ADVANCE PTR TO #                                                 4134
194B  7E                 MOV   A,M  }                                                             4135
194C  89                 ADC   C    } ADD HI BYTE OF # TO C                                       4136
194D  4F                 MOV   C,A  }                                                             4137
194E  C9                 RET                                                                      4138 CONVERT MANTISSA TO A POSITIVE #
194F  21 16 04      X194F  LXI   H,X0416 LOAD HL WITH ADDR OF BYTE CONTAINING SIGN OF RESULT (see 1B73)  4139 = TO MAGNITUDE OF RESULT
1952  7E                 MOV   A,M  } COMPLEMENT SIGN BIT OF RESULT
```

ADD A 3 BYTE # STORED IN MEMORY TO C D E REG'S

NOTE: CARRY BIT IS USED

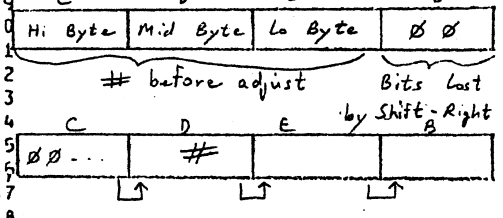
SIGN OF ARG IN HOLDING AREA

```
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
1959 79          MOV  A,L
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

                 COMPUTE   ØØ  ØØ  ØØ  ØØ
                          -C   -D   -E   -B

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 ADJUSTING #
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     }
1996 38              DATA A'8'  }   = 2.88539
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
                                  }
```

4140 /
4141
4142 /
4143
4144
4145 G
4146
4147
4148 _'
4149
4150
4151 W
4152
4153
4154 O
4155 I
4156
4157 V
4158 ZFINISH
4159 C
4160 Z
4161 G
4162
4163 C
4164 FCTR FIRST)
4165
4166 /
4167 -
4168 H
4169
4170
4171 O
4172
4173
4174 W
4175
4176
4177 -
4178
4179
4180 G
4181 C
4182
4183
4184
4185
4186
4187
4188 V
4189
4190
4191
4192 "
4193 E
4194 *
4195 8
4196
4197
4198
4199 !

Shift # in CDE Right n places

n in ACC



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 }
19A0  01 35 80           LXI  B,X8035 } LOAD BCDE WITH SNG-PREC CONSTANT = √2/2      5
19A3  11 F3 04           LXI  D,X04F3 }
19A6  90                 SUB  B COMPUTE POWER OF 2 REQUIRED TO ADJUST ARG TO THE RANGE .5 ≤ ARG < 1
19A7  F5                 PUSH PSW SAVE ARG MULTIPLIER ON STACK
19A8  70                 MOV  M,B FORCE ARG TO THE RANGE .5 ≤ ARG < 1 BY FORCING EXPONENT TO 80 (= 2^-1)
19A9  D5                 PUSH D } PUT CONSTANT = √2/2 ON STACK                       U
19AA  C5                 PUSH B }                                                    E
19AB  CD 9C 18           CALL X189C COMPUTE ARG + √2/2                               M
19AE  C1                 POP  B } PUT CONSTANT = √2/2 IN BCDE FROM STACK             A
19AF  D1                 POP  D }                                                    Q
19B0  04                 INR  B MULTIPLY CONSTANT BY 2 (CONSTANT NOW = √2)
19B1  CD 2E 1A           CALL X1A2E COMPUTE √2/(ARG + √2/2)                          M.
19B4  21 B7 19           LXI  H,X19B7 LOAD HL WITH ADDR OF CONSTANT = 1              !
19B7  CD 96 18           CALL X1896 COMPUTE 1-[√2/(ARG + √2/2)] = (ARG - √2/2)/(ARG + √2/2)   M
19BA  21 BB 19           LXI  H,X19BB LOAD HL WITH ADDR OF POLYNOMIAL TABLE FOR LOG  !
19BD  CD 58 26           CALL X2658 EVALUATE POLYNOMIAL                              MX&
19C0  01 80 80           LXI  B,X8080 } LOAD BCDE WITH SNG-PREC CONSTANT = -.5
19C3  11 00 00           LXI  D,X0000 }
19C6  CD 9C 18           CALL X189C ADD CONSTANT TO POLYNOMIAL                       M
19C9  F1                 POP  PSW FETCH ARG MULTIPLIER FROM STACK
19CA  CD 4C 21           CALL X214C CONVERT ARG MULTIPLIER TO SNG-PREC, ADD TO POLYNOMIAL   ML!
19CD  01 31 80    X19CD  LXI  B,X8031 } LOAD BCDE WITH SNG-PREC CONSTANT = .693147 = LN 2   1
19D0  11 18 72           LXI  D,X7218 } TO FINISH LOG, MULTIPLY POLYNOMIAL BY LN 2
-------------------------------------------------------------------------------------------------
19D3  EF          X19D3  RST  5 PERFORM FLOATING-PT SIGN TEST ON # IN HOLDING AREA
19D4  C8                 RZ   IF # IS ZERO, THEN MULTIPLY IS DONE & RESULT IS ALREADY IN HOLDING AREA   H
19D5  2E 00              MVI  L,H'00' SET L FOR MULTIPLY ROUTINE (SUM OF EXPONENTS)   .
19D7  CD AB 1A           CALL X1AAB COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS   M+
19DA  79                 MOV  A,C } STORE HI-8 BITS OF MULTIPLICAND IN ACI INSTRUCTION AT 1A09
19DB  32 0A 1A           STA  X1A0A }                                                2
19DE  EB                 XCHG      } STORE LO-16 BITS OF MULTIPLICAND IN LXI INSTRUCTION AT 1A04
19DF  22 05 1A           SHLD X1A05 }                                                "
19E2  01 00 00           LXI  B,X0000 } USE CDEB TO ACCUMULATE MANTISSA OF PRODUCT
19E5  50                 MOV  D,B     } INITIALLY SET ACCUMULATED VALUE TO 00 00 00 00   P
19E6  58                 MOV  E,B     } (B IS LOST-BIT BYTE)                         X
19E7  21 ED 18           LXI  H,X18ED } PUT ADDR OF (ROUTINE TO NORMALIZE # IN CDEB, ADJUST EXPONENT,   !
19EA  E5                 PUSH H       } THEN PUT RESULT IN HOLDING AREA) ON STACK - LAST THING TO DO AT END OF MULTIPLY
19EB  21 F3 19           LXI  H,X19F3 } PUT ADDR 19F3 ON STACK (TOTAL-LOOP 3 TIMES - 3 BYTES IN MULTIPLIER)   !
19EE  E5                 PUSH H       }
19EF  E5                 PUSH H       }
19F0  21 12 04           LXI  H,X0412 INITIALLY SET HL TO POINT TO LO-ORDER BYTE OF THE MULTIPLIER IN HOLDING AREA   !
19F3  7E          X19F3  MOV  A,M FETCH BYTE OF MULTIPLIER INTO ACC
19F4  23                 INX  H ADVANCE PTR TO NEXT BYTE IN MULTIPLIER               #
19F5  B7                 ORA  A TEST MULTIPLIER BYTE                                 7
19F6  CA 1E 1A           JZ   X1A1E IF MULTIPLIER BYTE IS 00, THEN NOTHING TO ACCUMULATE IN PRODUCT, SHIFT ACCUMULATED PRODUCT 8 BITS RIGHT
19F9  E5                 PUSH H SAVE PTR TO MULTIPLIER ON STACK
19FA  EB                 XCHG PUT 2-LO ORDER BYTES OF ACCUMULATED PRODUCT IN HL (SPEEDS UP BIT SHIFT ROUTINE)
19FB  1E 08              MVI  E,H'08' SET BIT-LOOP CTR TO 8 (8 BITS OF MULTIPLIER TO TEST)
19FD  1F          X19FD  RAR  ROTATE LO-BIT OF MULTIPLIER INTO THE CARRY BIT
19FE  57                 MOV  D,A SAVE REST OF MULTIPLIER BYTE IN D                  W
19FF  79                 MOV  A,C PUT HI-ORDER BYTE OF ACCUMULATED PRODUCT IN ACC FOR RIGHT BIT-SHIFT
1A00  D2 08 1A           JNC  X1A08 IF MULTIPLIER BIT WAS 0, DON'T ADD MULTIPLICAND TO ACCUMULATED PRODUCT
1A03  D5                 PUSH D SAVE REST OF MULTIPLIER BYTE & BIT-LOOP CTR ON STACK   U
1A04  11 00 00           LXI  D,X0000 LOAD DE WITH LO-16 BITS OF MULTIPLICAND
1A07  19                 DAD  D ADD LO-16 BITS OF MULTIPLICAND TO ACCUMULATED PRODUCT
1A08  D1                 POP  D FETCH REST OF MULTIPLIER BYTE INTO D, BIT-LOOP CTR INTO E FROM STACK
1A09  CE 00              ACI  H'00' ADD HI-8 BITS OF MULTIPLICAND TO ACCUMULATED PRODUCT   N
1A0B  1F          X1A0B  RAR  } SHIFT 3 BYTES OF ACCUMULATED PRODUCT & LOST-BIT BYTE
1A0C  4F                 MOV  C,A } RIGHT 1 PLACE                                    O
1A0D  7C                 MOV  A,H }
```

SINGLE-PRECISION MULTIPLY
(# IN BCDE) × (# IN HOLDING AREA)
MULTIPLICAND (1ST #) MULTIPLIER (2ND #)

C D E B
C H L B (AFTER XCHG 19FA)
LO-16 BITS AFTER LXI 1A04, DAD 1A07
CY H L
HI-8 BITS AFTER ACI 1A09

```
                                                                                        
1A0E  1F                   RAR           SHIFT 3 BYTES OF ACCUMULATED PRODUCT + LOST-BIT
1A0F  67                   MOV  H,A      BYTE RIGHT 1 PLACE
1A10  70                   MOV  A,L
1A11  1F                   RAR
1A12  6F                   MOV  L,A
1A13  78                   MOV  A,B
1A14  1F                   RAR                                                          G
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               B
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 IN CDE, LOST-BIT BYTE IS B
1A1C  E1          X1A1C    POP  H FETCH PTR TO MULTIPLIER FROM STACK                    I
1A1D  C9                   RET LOOP TO 19F3 TWO TIMES, THEN JMP NORMALIZE # ROUTINE 18ED, DONE AFTER NORMALIZE   C
-----------------------------------------------------------------------------------------------
1A1E  43          X1A1E    MOV  B,E    8 BIT SHIFT RIGHT                                Z    PERFORM 8 BIT SHIFT RIGHT ON
1A1F  5A                   MOV  E,D      ACC                                            O    # IN CDE, B IS LOST-BIT BYTE
1A20  51                   MOV  D,C      ØØ → C → D → E → B                             O
1A21  4F                   MOV  C,A                                                     I
1A22  C9                   RET                                                          M6
-----------------------------------------------------------------------------------------------
1A23  CD 36 1B    X1A23    CALL X1B36 MOVE SNG-PREC # FROM HOLDING AREA ONTO STACK (1ST #)  !<
1A26  21 8C 1F             LXI  H,X1F8C LOAD HL WITH ADDR OF SNG-PREC CONSTANT IS DECIMAL } 2ND #   MC
1A29  CD 43 1B             CALL X1B43 MOVE CONSTANT INTO HOLDING AREA                   A
1A2C  C1          X1A2C    POP  B } MOVE FLOATING-PT # FROM STACK INTO BCDE, B = EXPONENT   Q
1A2D  D1                   POP  D }   (1ST #)
-----------------------------------------------------------------------------------------------
1A2E  EF          X1A2E    RST  5   } TEST DIVISOR STORED IN HOLDING AREA                    SINGLE-PRECISION DIVIDE
1A2F  CA AC 04             JZ   X04AC } IF DIVISOR IS ZERO, PRINT ERROR MESSAGE "DIVISION BY ZERO"  .   (# IN BCDE) ÷ (# IN HOLDING AREA)
1A32  2E FF                MVI  L,H'FF' SET L FOR DIVIDE COMPUTATION (DIFFERENCE OF EXPONENTS)  H+   (1ST #)       (2ND #)
1A34  CD AB 1A             CALL X1AAB COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS  4
1A37  34                   INR  M } ADD TWO TO EXPONENT OF RESULT (LOC Ø415)            4
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    2}
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   2Y
1A40  32 59 1A             STA  X1A59 }                                                 +
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   2U
1A45  32 55 1A             STA  X1A55 }                                                 A
1A48  41                   MOV  B,C } DIVIDEND MANTISSA ORIGINALLY IN CDE
1A49  EB                   XCHG     } MOVE MANTISSA TO BHL.                             /
1A4A  AF                   XRA  A   CLEAR ACC                                           C
1A4B  4F                   MOV  C,A } CLEAR CDE - USED TO ACCUMULATE QUOTIENT           K
1A4C  57                   MOV  D,A }
1A4D  5F                   MOV  E,A }
1A4E  32 60 1A             STA  X1A60 CLEAR 1A6Ø - EXTRA BYTE OF DIVIDEND AT HI END - ie DIVIDEND IS IN 1A6Ø B H L
1A51  E5          →X1A51   PUSH H } PUT DIVIDEND ON STACK (DIVIDEND IN BHL) (WHAT'S LEFT OF IT AT THIS POINT)   E
1A52  C5                   PUSH B }
1A53  7D                   MOV  A,L    }                                                V
1A54  D6 00                SUI  H'00'  } LO-BYTE     COMPUTE DIVIDEND = DIVIDEND - DIVISOR
1A56  6F                   MOV  L,A    }                     (in BHL)   (in BHL)  (in 1A5D 1A59 1A55)
1A57  7C                   MOV  A,H    }                                                ^
1A58  DE 00                SBI  H'00'  } MID-BYTE    IF NC, DIVIDEND WAS ≥ DIVISOR - SUB OK, PUT A 1 IN QUOTIENT
1A5A  67                   MOV  H,A    }
1A5B  78                   MOV  A,B    }             IF C, DIVIDEND WAS < DIVISOR - SUB NOT OK, PUT A Ø IN QUOTIENT
1A5C  DE 00                SBI  H'00'  } HI-BYTE                                        G
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)
```

```
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 THE
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 MANTISSA (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,XØ415 } 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,XØ415 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
```

```
4320
4321
4322
4323
4324
4325
4326  1A5J - 1A5E
4327
4328
4329
4330  ROUNDOFF TEST AT 1926-1927
4331  NORMALIZATION, AND PUT SIGN BIT INTO RESULT
4332  BYTES
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359  (# IN Ø4ØE - Ø415) × ÷ (# IN Ø418 - Ø41F)
4360  (1ST #)  (2ND #)
4361
4362
4363
4364
4365
4366  COMPUTE EXPONENT FOR MULTIPLY (L=Ø)
4367  OR DIVIDE (L=FF)
4368  (FORCE VALUE IN HOLDING AREA TO Ø)
4369
4370
4371
4372
4373  FOR DIV E1-E2-1)
4374
4375
4376
4377
4378
4379
```

SHIFT CARRY BIT INTO 3 BYTES OF QUOTIENT
IF SUBTRACT OK, CARRY IS SET (see 1A6C)
OTHERWISE SHIFT A Ø INTO QUOTIENT

```
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 F1               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 02 19         JP    X1902 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                - OVERFLOW
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 10 X # IN HOLDING AREA                        4    4401
1AE7 C0               RNZ   IF EXPONENT BYTE NOT ZERO, MULTIPLY IS OK & FINISHED          @    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             @    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 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 CA 18         JMP   X18CA NORMALIZE # (INVERT SIGN BIT & 2's COMPLEMENT # IN CDEB IF SIGN  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 1B         JM    X1B1C NEGATE INTEGER VALUE IN HOLDING AREA (CONVERT # TO POSITIVE VALUE)  4424
1B10 CA 1A 1C         JZ    X1C1A 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 20 1C         JMP   X1C20 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
```

```
1B2C  2A 12 04           LHLD X0412 LOAD HL WITH INTEGER VALUE IN HOLDING AREA
1B2F  7C         X1B2F   MOV  A,H  } TEST IF INTEGER VALUE IS ØØØØ
1B30  B5                 ORA  L    }   +
1B31  C8                 RZ        } RETURN IF IT IS ØØØØ (ACC = ØØ)
1B32  7C                 MOV  A,H  PUT HI-ORDER BYTE OF INTEGER IN ACC
1B33  C3 F0 1A           JMP  X1AF0 SET FLAGS ACCORDING TO SIGN BIT OF BYTE IN ACC (ACC=FF IF MINUS, ACC=1 IF POSITIVE)
-----------------------------------------------------------------------------
1B36  EB         X1B36   XCHG       SAVE HL IN DE
1B37  2A 12 04           LHLD X0412 } PUT 4-BYTE SINGLE-PRECISION # IN HOLDING AREA AT
1B3A  E3                 XTHL       } Ø412 ON STACK, LSB FIRST, EXPONENT BYTE LAST
1B3B  E5                 PUSH H     }
1B3C  2A 14 04           LHLD X0414 }
1B3F  E3                 XTHL       }
1B40  E5                 PUSH H     }
1B41  EB                 XCHG       RESTORE HL FROM DE
1B42  C9                 RET
-----------------------------------------------------------------------------
1B43  CD 54 1B   X1B43   CALL X1B54 FETCH 4 BYTE # FROM MEMORY (HL IS PTR) INTO BCDE
1B46  EB         X1B46   XCHG       PUT PTR TO CONSTANT IN MEMORY IN DE, PUT MED + LO BYTES (DE) IN HL
1B47  22 12 04           SHLD X0412 STORE MED + LO BYTES OF #  } STORE 4 BYTE VALUE
1B4A  60                 MOV  H,B  } PUT HI BYTE (C) IN L      } IN HOLDING AREA
1B4B  69                 MOV  L,C  }                           }
1B4C  22 14 04           SHLD X0414 STORE HI BYTE + EXP BYTE   }
1B4F  EB                 XCHG       RESTORE PTR TO CONSTANT IN MEMORY TO HL, DE DON'T CARE
1B50  C9                 RET
-----------------------------------------------------------------------------
1B51  21 12 04   X1B51   LXI  H,X0412 LOAD HL WITH ADDR OF SNG-PREC OR INTEGER HOLDING AREA
1B54  5E         X1B54   MOV  E,M  FETCH VALUE IN HOLDING AREA INTO BCDE, B = EXPONENT BYTE
1B55  23                 INX  H    IF INTEGER, VALUE IN DE, BC DON'T CARE
1B56  56         X1B56   MOV  D,M  } IF ENTER 1B56, MOVE 3 BYTES FROM MEMORY INTO BCD
1B57  23                 INX  H    } (USED BY STRING FCN'S + STRING VARIABLE PROCESSING)
1B58  4E                 MOV  C,M  }
1B59  23                 INX  H    }
1B5A  46                 MOV  B,M  }
1B5B  23         X1B5B   INX  H
1B5C  C9                 RET
-----------------------------------------------------------------------------
1B5D  11 12 04   X1B5D   LXI  D,X0412 LOAD DE WITH ADDR OF HOLDING AREA (SOURCE PTR), HL HAS DEST. ADDR
1B60  06 04              MVI  B,H'04' SET # OF BYTES TO MOVE TO SNG-PREC VALUE
1B62  C3 6A 1B       +-- JMP  X1B6A MOVE THE # (4 BYTES)
1B65  EB         X1B65|  XCHG       EXCHANGE HL + DE - SOURCE + DESTINATION PTRS SWAPPED
1B66  3A A4 03   X1B66|  LDA  X03A4 } LOAD # OF BYTES TO MOVE INTO B REG
1B69  47              |  MOV  B,A   } (VALUE TYPE = # OF BYTES IN VALUE)
1B6A  1A       ->X1B6A   LDAX D     FETCH A BYTE
1B6B  77              |  MOV  M,A   STORE BYTE
1B6C  13              |  INX  D  } ADVANCE PTRS
1B6D  23              |  INX  H  }
1B6E  05              |  DCR  B     DECREMENT CHAR CTR
1B6F  C2 6A 1B       +-- JNZ  X1B6A LOOP TIL ALL BYTES HAVE BEEN MOVED
1B72  C9                 RET
-----------------------------------------------------------------------------
1B73  21 14 04   X1B73   LXI  H,X0414 LOAD HL WITH ADDR OF MOST SIGNIFICANT BYTE OF VALUE IN HOLDING AREA
1B76  7E                 MOV  A,M  FETCH MOST SIGNIFICANT BYTE INTO ACC
1B77  07                 RLC  } FORCE SIGN BIT OF MANTISSA TO 1 } EXTRACT SIGN BIT, RESTORE
1B78  37                 STC  } (i ... s)                        } LEADING 1 IN MANTISSA
1B79  1F                 RAR                                       S, 1XXX XXXX -> Ø414
1B7A  77                 MOV  M,A  PUT MOST SIGNIFICANT BYTE BACK IN HOLDING AREA
1B7B  3F                 CMC       COMPLEMENT SIGN BIT
1B7C  1F                 RAR       PUT SIGN BIT-bar IN MSB      X S-bar, 1XX XXXX
1B7D  23                 INX  H  } STORE THIS SIGN BYTE AT LOC Ø416
1B7E  23                 INX  H  }
1B7F  77                 MOV  M,A}
1B80  79                 MOV  A,C FETCH MOST SIGNIFICANT BYTE OF 2ND # INTO ACC
1B81  07                 RLC } EXTRACT SIGN BIT, RESTORE LEADING 1 IN MANTISSA
1B82  37                 STC }
```

SET FLAGS + ACC FOR SIGN OF INTEGER VALUE

SINGLE-PREC # ONTO STACK — HOLDING AREA TO STACK

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

FETCH INT OR SNG-PREC VALUE FROM MEMORY. STORE IT IN HOLDING AREA

Ø412	(E)	LO BYTE	
Ø413	(D)	MED BYTE	
Ø414	(C)	HI BYTE	
Ø415	(B)	EXPONENT BYTE	

LOAD BCDE, OR BCD WITH A VARIABLE VALUE

Ø412 (E)
Ø413 (D)
Ø414 (C)
Ø415 (B)

MOVE N-BYTES (VARIABLE VALUE)

MOVE N CHAR:	1B66	1B65
SOURCE ADDR:	DE	HL
DEST ADDR:	HL	DE

(Ø3A4): # OF BYTES TO MOVE

EXTRACT SIGN BITS FROM 2 #'S, RESTORE LEADING 1 IN BOTH #'S (ONE # IN HOLDING AREA, ONE # IN BCDE)

SUITABLE FOR SNG- + DBL-PREC

ACC HAS XNOR OF SIGN BITS

Ø416 HAS SIGN-bar OF # IN HOLDING AREA

```
                        RAR   } EXTRACT SIGN BIT, RESTORE LEADING 1 IN MANTISSA  S2 1XXX XXXX
1883  1F                MOV   C,A PUT MOST SIGNIFICANT BYTE BACK INTO REG C                  0
1884  4F                RAR   PUT SIGN BIT IN MSB      X S2 1XX XXXX
1885  1F                XRA   M FORM XOR OF SIGN BITS IN ACC  S1 ⊕ S2 in MSB of ACC
1886  AE                RET                                                                .
1887  C9                                                                                   I

1888  21 18 04  X1888   LXI   H,X0418 LOAD HL WITH ADDR OF LSB OF DBL-PREC # IN TEMP STORAGE AREA
1888  11 65 18  X1888   LXI   D,X1865 LOAD DE WITH ADDR OF ROUTINE THAT WILL MOVE DBL-PREC #
188E  C3 97 18          JMP   X1897 LOAD DE WITH Ø4ØE, THEN MOVE THE VARIABLE VALUE          C
1891  21 18 04  X1891   LXI   H,X0418 LOAD HL WITH ADDR OF LSB OF DBL-PREC # IN TEMP STORAGE AREA
1894  11 66 18          LXI   D,X1866 LOAD DE WITH ADDR OF ROUTINE THAT WILL MOVE DBL-PREC # FROM Ø4ØE
1897  D5          X1897  PUSH  D PUT IMPLIED JMP ADDR ON STACK                              U
1898  11 12 04  X1898   LXI   D,X0412 LOAD DE WITH ADDR OF LSB OF # IN HOLDING AREA (INT OR SNG)
189B  F7                RST   6       TEST VALUE TYPE
189C  D8                RC            RETURN IF INT, SNG, OR STRING TYPE                   X
189D  11 0E 04          LXI   D,X040E LOAD DE WITH ADDR OF LSB OF # IN DBL PREC HOLDING AREA
18A0  C9                RET                                                                I

18A1  78          X18A1  MOV   A,B } TEST EXPONENT BYTE
18A2  B7                 ORA   A  }                                                         7
18A3  CA 28 00           JZ    X0028 IF EXPONENT IS ZERO, THEN # IN BCDE IS ZERO, DO FLOATING-PT
18A6  21 EF 1A           LXI   H,X1AEF } PUT ADDR 1AEF ON STACK FOR IMPLIED JMP
18A9  E5                 PUSH  H       } (JMP TO ROUTINE TO TEST SIGN BASED ON COMPLEMENT OF BYTE IN ACC)
18AA  EF                 RST   5 TEST SIGN OF FLOATING-PT # IN HOLDING AREA
18AB  79                 MOV   A,C PUT MSB BYTE OF # IN BCDE IN ACC
18AC  C8                 RZ    IF # IN HOLDING AREA IS ZERO, INVERT SIGN BIT, DO SIGN TEST   H
18AD  21 14 04           LXI   H,X0414 LOAD HL WITH ADDR OF MSB BYTE OF # IN HOLDING AREA
18B0  AE                 XRA   M XOR SIGN BITS OF THE 2 #'S                                 .
18B1  79                 MOV   A,C PUT MSB OF REGISTER # IN ACC
18B2  F8                 RM    IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS COMPLEMENTED SIGN OF SUBTRAHEND
18B3  CD B9 18           CALL  X18B9 TEST THE RELATIVE SIZE OF THE 2 #'S, RC (M)>(R), RNC (M)<(R)   9
18B6  1F          X18B6  RAR   PUT CARRY BIT IN ACC  } VERY TRICKY! NOTE: SIGN BITS SAME FOR CALL 18B9
18B7  A9                 XRA   C XOR WITH SIGN OF (R #) } REALLY REVERSED DURING MAGNITUDE TEST
18B8  C9                 RET                            } BASED ON MAGNITUDE OPPOSITE THAT IF   I

18B9  23          X18B9  INX   H  ADVANCE HL TO POINT TO EXPONENT BYTE                      #
18BA  78                 MOV   A,B PUT EXPONENT-BYTE OF REGISTER # IN ACC                   >
18BB  BE                 CMP   M  TEST AGAINST EXPONENT BYTE OF # IN MEMORY                 @
18BC  C0                 RNZ      RETURN IF NOT EQUAL                                     +
18BD  2B                 DCX   H  } TEST MSB-BYTE, RETURN IF NOT EQUAL
18BE  79                 MOV   A,C }
18BF  BE                 CMP   M   }                                                        >
18C0  C0                 RNZ       }                                                        @
18C1  2B                 DCX   H  }                                                         +
18C2  7A                 MOV   A,D } TEST MID-BYTE, RETURN IF NOT EQUAL
18C3  BE                 CMP   M  }
18C4  C0                 RNZ      }                                                        >
18C5  2B                 DCX   H  }                                                        @
18C6  7B                 MOV   A,E } TEST LO-BYTE, RETURN IF NOT EQUAL
18C7  96                 SUB   M  }
18C8  C0                 RNZ      }                                                        @
18C9  E1                 POP   H  REMOVE ADDR 18B6 FROM STACK } IF BOTH #'S ARE EQUAL, ZERO
18CA  E1          X18CA  POP   H  REMOVE ADDR 1AEF FROM STACK } FLAG SET, RETURN DIRECTLY TO
18CB  C9                 RET                                 } CALLING ROUTINE ONCE REMOVED  I

18CC  7A          X18CC  MOV   A,D } XOR SIGN BITS OF THE 2 #'S
18CD  AC                 XRA   H  }                                                         ,
18CE  7C                 MOV   A,H }
18CF  FA F0 1A           JM    X1AF0 IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS SIGN OF MINUEND (#HL)
18D2  BA                 CMP   D  COMPARE HI-ORDER BYTES (H-D)                              !
18D3  C2 F1 1A           JNZ   X1AF1 IF HI-ORDER BYTES ARE UNEQUAL, SET FLAGS ACCORDINGLY   @
18D6  7C                 MOV   A,L } COMPARE LO-ORDER BYTES
18D7  93                 SUB   E  }
18D8  C2 F1 1A           JNZ   X1AF1 IF LO-ORDER BYTES ARE UNEQUAL, SET FLAGS ACCORDINGLY   8
```

1883 4500
1884 4501
1885 4502
1886 4503
1887 4504
1888 4505
1888 4506 FROM Ø418 TO Ø4ØE
188E 4507
1891 4508
1894 4509 TO Ø418
1897 4510 ROUTINE:
1898 4511 LOAD DE WITH LSB ADDR OF #
4512
4513
4514
4515
4516
4517
4518 TEST ON # IN HOLDING AREA
4519
4520 COMPUTE SIGN OF SNG-PREC
4521 FLOATING-PT SUBTRACT
4522 (# IN MEMORY) - (# IN BCDE)
4523
4524 MINUEND SUBTRAHEND
4525
4526
4527
4528
4529
4530
4531
4532
4533 TEST IF SNG-PREC # IN
4534 BCDE EQUALS SNG-PREC #
4535 IN HOLDING AREA (Ø412 – Ø415)
4536
4537
4538 IF BOTH #'S ARE SAME SIGN,
4539
4540 THEN RC # IN MEM > # IN REG
4541 RNC # IN MEM ≤ # IN REG
4542 RZ # IN MEM = # IN REG
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559

1888 LOAD DE WITH ADDR OF ROUTINE THAT WILL MOVE DBL-PREC # FROM Ø418 TO Ø4ØE

ROUTINE: MOVE #
1888 DBL-PREC Ø418 → Ø4ØE
1888 # (HL) → HOLDING AREA
1891 DBL-PREC Ø4ØE → Ø418
1897

(#M) - (#R) SIGN TEST (SNG-PREC MAGNITUDE TEST)

2 FLOATING-POINT #'S
RZ (#M) = (#R) ACC = Ø RC (#M) < (#R) ACC = -1
RNC (#M) ≥ (#R) ACC = 1

IF #'S ARE MINUS, MAGNITUDES ARE (#R) IS MINUS, SIGN OF SUBTRACT IS POSITIVE

(12 CALL BEFORE CALL)

(#HL) - (#DE) SIGN TEST
(INTEGER MAGNITUDE TEST)

RZ (#HL) = (#DE) ACC = Ø → =
RNC (#HL) ≥ (#DE) ACC = 1 → >
RC (#HL) < (#DE) ACC = -1 → <

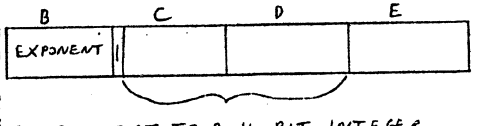
```
1B0B  C9                  RET RETURN WITH ZERO-FLAG SET IF #'S ARE EQU.     ACC = Ø                          I
1B0C  21 18 04   X1B0C    LXI  H,X0418 LOAD HL WITH ADDR OF DBL-PREC VALUE TEMP STORAGE AREA                 !
1B0F  CD 66 1B            CALL X1B66 MOVE DBL-PREC # SOURCE ADDR IN DE, DESTINATION ADDR IN HL               M
1BE2  11 1F 04   X1BE2    LXI  D,X041F LOAD DE WITH ADDR OF EXPONENT BYTE OF 2ND #                                 (# Ø40E-Ø415) - (# Ø418-Ø41F) SIGN TEST
1BE5  1A                  LDAX D } PUT EXPONENT BYTE IN ACC + TEST IT                                              DBL-PREC MAGNITUDE TEST
1BE6  B7                  ORA  A }           FLTING-PT SIGN TEST ON # IN HOLDING AREA
1BE7  CA 28 00            JZ   X0028 IF EXPONENT IS ZERO, THEN # IN TEMP STORAGE AREA IS ZERO, DO           J(     MINUEND        SUBTRAHEND
1BEA  21 EF 1A            LXI  H,X1AEF } PUT ADDR 1AEF ON STACK FOR IMPLIED JMP                               !
1BEC  E5                  PUSH H       } (JMP TO ROUTINE TO TEST SIGN BASED ON COMPLEMENT OF BYTE IN ACC)
1BEE  EF                  RST  5 TEST SIGN OF FLOATING-PT # IN HOLDING AREA
1BEF  1B                  DCX  D BACK UP PTR TO # IN TEMP STORAGE (POINT TO SIGN BYTE)
1BF0  1A                  LDAX D LOAD MSB INTO ACC
1BF1  4F                  MOV  C,A SAVE MSB OF # IN TEMP STORAGE (2ND #) IN REG C                      O
1BF2  C8                  RZ IF # IN HOLDING AREA IS ZERO (1ST #), INVERT SIGN BIT OF 2ND #, DO SIGN TEST   H
1BF3  21 14 04            LXI  H,X0414 LOAD HL WITH ADDR OF SIGN BYTE OF # IN HOLDING AREA (1ST #)    !
1BF6  AE                  XRA  M XOR SIGN BITS OF THE 2 #'S                                            .
1BF7  79                  MOV  A,C PUT MSB OF 2ND # IN ACC
1BF8  F8                  RM IF SIGN BITS ARE OPPOSITE, RESULT OF SUB IS COMPLEMENTED SIGN OF SUBTRAHEND (2ND #)
1BF9  13                  INX  D BACK UP PTR TO 2ND # (POINT TO EXPONENT BYTE)
1BFA  23                  INX  H BACK UP PTR TO 1ST # (POINT TO EXPONENT BYTE)                        #
1BFB  06 08               MVI  B,H'08' SET LOOP CTR TO 8 (# OF BYTES IN DBL-PREC #)
1BFD  1A         X1BFD    LDAX D FETCH A BYTE FROM 2ND #
1BFE  96                  SUB  M SUBTRACT A BYTE FROM 1ST #
1BFF  C2 B6 1B            JNZ  X1BB6 IF #'S ARE UNEQUAL, PUT CARRY BIT IN ACC, XOR WITH SIGN BIT OF 2ND #  B6    (SEE 1BA1 SNG-PREC ROUTINE)
1C02  1B                  DCX  D DECREMENT PTR TO 2ND #
1C03  2B                  DCX  H DECREMENT PTR TO 1ST #                                                +
1C04  05                  DCR  B DECREMENT LOOP CTR
1C05  C2 FD 1B            JNZ  X1BFD LOOP UNTIL ALL 8 BYTES HAVE BEEN COMPARED                         B
1C08  C1                  POP  B REMOVE ADDR 1AEF FROM STACK - IMPLIED JMP NO LONGER NEEDED            A
1C09  C9                  RET RZ ACC = Ø, #'S ARE EQUAL                                                I
1C0A  CD E2 1B            CALL X1BE2 TEST RELATIVE MAGNITUDE OF #'S                                    M     DBL MAGNITUDE TEST
1C0D  C2 EF 1A            JNZ  X1AEF JMP TO SET FLAGS + ACC, RNC (# Ø40E-Ø415) > (# Ø418-Ø41F) ACC=1    B      (# Ø40E-Ø415) - (# Ø418-Ø41F)
1C10  C9                  RET RZ (# Ø40E-Ø415) = (# Ø418-Ø41F) ACC=Ø   RC (# Ø40E-Ø415) < (# Ø418-Ø41F) ACC=-1
1C11  F7         X1C11    RST  6      TEST TYPE OF VALUE IN HOLDING AREA
1C12  2A 12 04            LHLD X0412  LOAD HL WITH INTEGER VALUE FROM HOLDING AREA                     *     CINT
1C15  F8                  RM          RETURN IF VALUE IS ALREADY INTEGER FORMAT
1C16  CA 3A 1C            JZ   X1C3A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"
1C19  D4 40 1C            CNC  X1C40 IF VALUE IS DBL-PREC, CONVERT TO SNG-PREC                         TM    CONVERT FLOATING-PT # TO INTEGER
1C1C  21 3E 19            LXI  H,X193E } PUT ADDR OF ROUTINE TO PRINT ERROR MESSAGE "OVERFLOW"         !>                                  FORMAT
1C1F  E5                  PUSH H       } ON STACK
1C20  3A 15 04   X1C20    LDA  X0415 LOAD ACC WITH EXPONENT BYTE OF # IN HOLDING AREA                  :
1C23  FE 90               CPI  H'90' TEST EXPONENT BYTE
1C25  D2 36 1C            JNC  X1C36 IF EXPONENT IS 15. OR LARGER, TEST IF SPECIAL CASE -32768. IF IT IS,   THEN PUT IT IN INTEGER FORMAT,
1C28  CD 8F 1C            CALL X1C8F CONVERT FLOATING-POINT VALUE TO INTEGER FORMAT                    M     OTHERWISE PRINT "OVERFLOW"
1C2B  EB                  XCHG PUT CONVERTED VALUE IN HL
1C2C  D1         X1C2C    POP  D REMOVE 1 ADDR FROM STACK                                              Q
1C2D  22 12 04   X1C2D    SHLD X0412 STORE INTEGER VALUE IN HOLDING AREA   FROM 2173  STORE # TO PRINT IN Ø412+Ø413
1C30  3E 02      X1C30    MVI  A,H'02' } STORE Ø2 IN Ø3A4           } SET Ø3A4 TO INTEGER FORMAT       >
1C32  32 A4 03   X1C32    STA  X03A4   } SET VALUE TYPE TO INTEGER                                     2$
1C35  C9                  RET  END OF VALUE TYPE CONVERSION                                            I
1C36  01 80 90   X1C36    LXI  B,X9080 } PUT SNG-PREC # = -32768 DECIMAL IN BCDE
1C39  11 00 00            LXI  D,X0000 }
1C3C  CD A1 1B            CALL X1BA1 TEST IF # IN HOLDING AREA IS = -32768                             M!
1C3F  C0          <-----  RNZ  IF # ≠ -32768, LEAVE IT IN SNG-PREC FORMAT                              a
1C40  61                  MOV  H,C } PUT 8000 IN HOLDING AREA (INTEGER FORMAT FOR -32768)
1C41  6A                  MOV  L,D }
1C42  C3 2C 1C            JMP  X1C2C STORE # INTO HOLDING AREA AS AN INTEGER                           C.
1C45  F7         X1C45    RST  6      TEST TYPE OF VALUE IN HOLDING AREA                                     CSGN
1C46  E0                  RPO         RETURN IF VALUE IS ALREADY SINGLE PRECISION FORMAT
1C47  FA 60 1C            JM   X1C60 IF VALUE IS INTEGER, CONVERT TO SINGLE PRECISION
```

```
1C4A  CA 8A 1C          JZ    X1C8A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"
1C4D  CD 51 18  X1C4D   CALL  X1851 PUT 4-BYTE SINGLE PRECISION VALUE IN BCDE, B = EXPONENT
1C50  CD 83 1C          CALL  X1C83 SET VALUE TYPE TO SINGLE PRECISION
1C53  78                MOV   A,B  } PUT EXPONENT BYTE IN ACC
1C54  B7                ORA   A    } TEST EXPONENT
1C55  C9                RZ    IF EXPONENT IS ZERO, VALUE IS ALSO EXACTLY ZERO, NO MORE CONVERSION
1C56  CC 73 18          CALL  X1873 RESTORE LEADING 1 IN # IN MSB, EXTRACTED SIGN BIT IN LOC Ø416
1C59  21 11 04          LXI   H,X0411 } USE Ø411 CONTENTS AS THE LOST BIT BYTE FOR ROUND-OFF
1C5C  46                MOV   B,M     }
1C5D  C3 22 19          JMP   X1922 PERFORM ROUNDOFF & NORMALIZE RESULT

1C60  2A 12 04  X1C60   LHLD  X0412 LOAD HL WITH 2-BYTE INTEGER VALUE FROM HOLDING AREA
1C63  CD 83 1C  X1C63   CALL  X1C83 SET VALUE TYPE TO SINGLE PRECISION
1C66  7C                MOV   A,H } PUT # TO CONVERT IN ACC & D REG
1C67  55                MOV   D,L }
1C68  1E 00             MVI   E,H'00' CLEAR REG E AS PART OF FLOATING-PT #
1C6A  06 90             MVI   B,H'90' SET ' ' VALUE FOR EXPONENT TO 9Ø (= 2^15 -> 9ØH-81H=F)
1C6C  C3 FA 1A          JMP   X1AFA CONVERT # IN BADE TO SNG-PREC (CONSIDER SIGN-BIT + USE NORMALIZE ROUTINE)

1C6F  F7        X1C6F   RST   6  TEST TYPE OF VALUE IN HOLDING AREA
1C70  D0                RNC      RETURN IF VALUE IS ALREADY DOUBLE PRECISION FORMAT
1C71  CA 8A 1C          JZ    X1C8A IF VALUE IS STRING TYPE, PRINT ERROR MESSAGE "TYPE MISMATCH"
1C74  FC 60 1C          CM    X1C60 IF VALUE IS INTEGER, CONVERT TO SINGLE PRECISION (1C6Ø)
1C77  21 00 00  X1C77   LXI   H,X0000 } ZERO LO. ORDER 4 BYTES OF DOUBLE-PRECISION
1C7A  22 0E 04          SHLD  X040E   } HOLDING AREA
1C7D  22 10 04          SHLD  X0410   }
1C80  3E 08             MVI   A,H'08' SET VALUE TYPE TO DOUBLE PRECISION
1C82  21 3E 04          LXI   H,X043E DUMMY MVI A,Ø4. - 1C83 SET VALUE TYPE TO SNG-PRECISION
1C85  C3 32 1C          JMP   X1C32   STORE VALUE TYPE IN Ø3A4, THEN RETURN

1C88  F7        X1C88   RST   6       TEST VARIABLE TYPE (OR VALUE TYPE IN HOLDING AREA)
1C89  C8                RZ            RETURN IF STRING VARIABLE
1C8A  1E 0D     X1C8A   MVI   E,H'0D' } PRINT ERROR MESSAGE "TYPE MISMATCH"
1C8C  C3 B7 04          JMP   X04B7   }

1C8F  47        X1C8F   MOV   B,A } EXPONENT BYTE IS IN ACC (VALUE IS IN HOLDING AREA)
1C90  4F                MOV   C,A } PUT ACC IN ALL 4 REG'S BCDE
1C91  57                MOV   D,A } IF EXPONENT IS Ø, VALUE IS Ø, ALL 4 BYTES OF # MUST BE Ø
1C92  5F                MOV   E,A }
1C93  B7                ORA   A  } TEST ACC - IF EXPONENT IS Ø, BCDE HAVE FLOATING-PT
1C94  C8                RZ       } VALUE = Ø  RETURN SINCE # IS A SPECIAL CASE INTEGER
1C95  E5                PUSH  H SAVE HL ON STACK
1C96  CD 51 18          CALL  X1851 PUT 4-BYTE SINGLE PRECISION VALUE IN BCDE FROM HOLDING AREA
1C99  CD 73 18          CALL  X1873 EXTRACT SIGN BIT (PUT IN Ø416), RESTORE LEADING 1 IN MANTISSA
1C9C  AE                XRA   M  PUT SIGN BIT OF # IN ACC          -> MANTISSA IS A POSITIVE # AFTER 1C99)
1C9D  67                MOV   H,A SAVE SIGN BIT OF # IN H
1C9E  FC B3 1C          CM    X1CB3 IF # IS NEGATIVE, DECREMENT MANTISSA BY 1. (LSB ROUNDING)
1CA1  3E 98             MVI   A,H'98' } COMPUTE (FROM EXPONENT BYTE IN B) # OF PLACES TO RIGHT SHIFT TO
1CA3  90                SUB   B       } LEAVE EQUIVALENT INTEGER VALUE IN DE (98 -> 2^23)
1CA4  CD 63 19          CALL  X1963 PERFORM RIGHT SHIFT, # OF PLACES TO SHIFT IN ACC
1CA7  7C                MOV   A,H PUT SIGN BYTE IN ACC (see 1C9D)
1CA8  17                RAL       TEST SIGN BYTE BY SHIFTING SIGN BIT INTO CARRY BIT
1CA9  DC 34 19          CC    X1934 ADD 1 TO RESULT - SINCE RESULT IS A POSITIVE MAGNITUDE AT THE MOMENT, THIS IS THE
1CAC  06 00             MVI   B,H'00' CLEAR B, THIS IS THE ACCUMULATED LOST-BIT REG, FORCE IT TO Ø FOR 194F
1CAE  DC 4F 19          CC    X194F NEGATE RESULT (INTEGER FORMAT # IN DE) BY SUBTRACTING # FROM Ø
1CB1  E1                POP   H RESTORE HL FROM THE STACK
1CB2  C9                RET

1CB3  1B        X1CB3   DCX   D DECREMENT LO-ORDER BYTE OF 16-BITS OF MANTISSA THAT WILL BECOME AN INTEGER - THIS
1CB4  7A                MOV   A,D } IF DE NOT FF, RETURN
1CB5  A3                ANA   E   }
1CB6  3C                INR   A   }
1CB7  C0                RNZ       }
1CB8  0B        X1C88   DCX   B ADJUST EXPONENT BYTE
1CB9  C9                RET
```

```
4620
4621
4622
4623   SNG-PREC
4624 Ø415 Ø414 Ø413 Ø412 Ø411
4625 EXP   MSB
4626
4627
4628
4629
4630 CONVERT INTEGER TO SINGLE
4631 PRECISION FORMAT
4632
4633 (INTEGER VALUE IS IN HL)
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651 CONVERT FLOATING-POINT VALUE
4652
4653 TO INTEGER #, FLOATING POINT
4654 FORMAT (REMOVE THE FRACTION
4655
4656 IN A FLOATING POINT #), RESULT
4657 IS INTEGER FORMAT # IN DE
4658
4659
4660
4661
4662
4663 TO CONVERT TO A 16 BIT INTEGER
4664 ONLY THE BITS IN
4665
4666
4667
4668 SAME AS TRUNCATE DOWNWARD, IE INT(-2.3)
4669                                  = -3
4670
4671
4672
4673 IS THE ROUND-DOWN FOR A NEGATIVE #
4674
4675
4676
4677
4678
4679
```

CDBL

STRING VALUE CHECK



104

```
1C9A  F7          RST  6     TEST TYPE OF VALUE IN HOLDING AREA
1C9B  F8          RM         RETURN IF VALUE IS ALREADY INTEGER TYPE VALUE
1C9C  D2 DE 1C    JNC  X1CDE IF VALUE IS DBL-PREC TYPE, JMP TO ROUTINE TO EXTRACT INTEGER PORTION OF #
1C9F  CA 8A 1C    JZ   X1C8A 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
1CC5  21 15 04 X1CC5 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  CC 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
1CF1  2B       X1CF1 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 ≥ 98 (≥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 1CFB)
1D03  CD 51 18    CALL X1851 MOVE HI-4 BYTES OF DBL-PREC # INTO BCDE (USED BY 1873 ROUTINE)
1D06  CD 73 18    CALL X1873 EXTRACT SIGN BIT (PUT IN 0416), RESTORE LEADING 1 IN MANTISSA (MANTISSA IS A POSITIVE # AFTER 1D06)
1D09  AE          XRA  M     PUT SIGN BIT OF # IN ACC
1D0A  2B          DCX  H     BACK-UP HL TO POINT TO LOC 0415
1D0B  36 9A       MVI  M,H'9A' 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 38       MVI  A,H'38' } 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 0F 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

```
1028  7E          X102A  MOV  A,M FETCH A BYTE FROM MANTISSA INTO ACC
1029  35                 DCR  M  DECREMENT THE MANTISSA (WHERE IT IS STORED)
102C  B7                 ORA  A  TEST THE MANTISSA BYTE
102D  23                 INX  H  ADVANCE PTR TO NEXT BYTE OF MANTISSA
102E  CA 2A 10           JZ   X102A IF BYTE WAS Ø, DECREMENTING PRODUCES A BORROW INTO NEXT HIGHER BYTE
1031  C9                 RET  DONE
-------------------------------------------------------------------------------------------------------
1032  E5          X1032  PUSH H  SAVE CONTENTS OF HL ON STACK
1033  21 A4 03           LXI  H,X03A4 LOAD HL WITH ADDR OF FLAG FOR TYPE OF VALUE IN HOLDING
1036  7E                 MOV  A,M FETCH VALUE TYPE INTO ACC
1037  36 02              MVI  M,H'02' FORCE VALUE TYPE TO 'INTEGER'
1039  F5                 PUSH PSW SAVE ORIGINAL VALUE-TYPE ON STACK
103A  60                 MOV  H,B } PUT # OF ELEMENTS IN DIMEN. IN HL
103B  69                 MOV  L,C }
103C  CD 75 10           CALL X1075 PERFORM INTEGER MULTIPLY (MULTIPLICAND IN HL, MULTIPLIER IN DE)
103F  CD 11 1C           CALL X1C11 CONVERT VALUE IN HOLDING AREA TO INTEGER FORMAT
1042  F1                 POP  PSW } RESTORE VALUE-TYPE FLAG TO ORIGINAL VALUE
1043  32 A4 03           STA  X03A4 }
1046  EB                 XCHG PUT VALUE IN DE
1047  F1                 POP  H RESTORE HL TO ORIGINAL VALUE FROM STACK
1048  C9                 RET
-------------------------------------------------------------------------------------------------------
1049  7C                 MOV  A,H PUT SIGN BYTE IN ACC (2ND #)
104A  17                 RAL    } TEST SIGN      - MINUS IF ACC = -1
104B  9F                 SBB  A } OF 2ND #        PLUS IF ACC = Ø
                  DONE THIS WAY
104C  47          FOR -32768 MOV  B,A SAVE SIGN OF 2ND # IN REG B
104D  CD 12 1E           CALL X1E12 NEGATE # IN HL (2ND #) BY SUBTRACTING FROM ØØØØ
1050  79                 MOV  A,C PUT ØØ IN ACC
1051  98                 SBB  B INVERT SIGN OF 2ND # (MINUS = -1, POS = Ø)
1052  C3 58 10           JMP  X1058
-------------------------------------------------------------------------------------------------------
1055  7C          X1055  MOV  A,H PUT SIGN BYTE IN ACC (2ND #)
1056  17                 RAL    } TEST SIGN      - MINUS IF ACC = -1
1057  9F                 SBB  A } OF 2ND #        PLUS IF ACC = Ø
1058  47          X1058  MOV  B,A SAVE SIGN OF 2ND # IN REG B
1059  E5                 PUSH H  PUT 2ND # ON STACK
105A  7A                 MOV  A,D PUT SIGN BYTE IN ACC (1ST #)
105B  17                 RAL    } TEST SIGN      - MINUS IF ACC = -1
105C  9F                 SBB  A } OF 1ST #        PLUS IF ACC = Ø
105D  19                 DAD  D ADD 1ST # TO 2ND # USING 16-BIT REGISTER ADD - RESULT IN HL
105E  88                 ADC  B } TEST FOR OVERFLOW (ADD PRODUCED A # TOO LARGE TO BE AN INTEGER
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
1064  C5                 PUSH B SAVE SIGN OF 2ND # ON STACK (MINUS IF ACC = -1  PLUS IF ACC = Ø)
1065  EB                 XCHG PUT 1ST # IN HL (IT WAS ORIGINALLY IN DE)
1066  CD 63 1C           CALL X1C63 CONVERT # IN HL TO SNG-PREC # IN HOLDING AREA
1069  F1                 POP  PSW PUT SIGN OF 2ND # IN ACC FROM STACK
106A  E1                 POP  H PUT 2ND # IN HL FROM STACK
106B  CD 36 1B           CALL X1B36 PUT SNG-PREC # IN HOLDING AREA ONTO STACK
106E  EB                 XCHG PUT 2ND # IN DE
106F  CD 2C 1E           CALL X1E2C DO AN ABBREVIATED CONVERSION TO SNG-PREC FORMAT
1072  C3 52 21           JMP  X2152 FETCH 1ST # FROM STACK INTO BCDE, DO A FLOATING-PT ADD
-------------------------------------------------------------------------------------------------------
1075  7C          X1075  MOV  A,H  } IF # IN HL IS ØØØØ, STORE INTEGER VALUE ØØØØ IN HOLDING AREA
1076  B5                 ORA  L    } SINCE PRODUCT WILL BE ØØØØ
1077  CA 20 1C           JZ   X1C20 }
107A  E5                 PUSH H  SAVE 1ST # ON STACK
107B  D5                 PUSH D  SAVE 2ND # ON STACK
107C  CD 06 1E           CALL X1E06 COMPUTE SIGN OF PRODUCT (IN B), CONVERT BOTH #'S TO POSITIVE
107F  C5                 PUSH B  SAVE BYTE CONTAINING CORRECT SIGN FOR PRODUCT ON STACK
1080  44                 MOV  B,H } PUT 1ST # IN BC
1081  4D                 MOV  C,L }
1082  21 00 00           LXI  H,X0000 LOAD HL WITH ØØØØ (START ACCUMULATED VALUE AT ØØØØ)
```

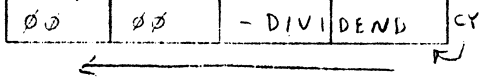
```
4740
4741
4742
4743
4744 LOOP UNTIL NON-ZERO BYTE FOUND
4745 (# IS NOT Ø - See ICE4, # HAS MSB-see 10Ø6)
4746
4747 MULTIPLY FOR DIM & ARRAYS
4748
4749
4750
4751
4752        (HL) ← (BC)
4753 1075   (HL) = (HL) * (DE)
4754
4755
4756
4757
4758        (DE) ← (HL)
4759
4760
4761 INTEGER SUBTRACT
4762 (# IN DE) - (# IN HL)
4763   (1ST #)  - (2ND #)
4764
4765
4766
4767
4768
4769 INTEGER ADDITION
4770 (# IN DE) + (# IN HL)
4771    1ST #      2ND #
4772
4773  ARGS    RESULT   CY    SIGN1+SIGN2+CY   XRA H
4774  + +       +      Ø          Ø            Ø
4775  + -       +      1          Ø            Ø
4776  - +       -      Ø          -1           Ø
4777  - -       -      1          -1           Ø
4778
4779
4780 RESULT (REMOVES 2 BYTES FROM STACK 1ST)
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791 INTEGER FORMAT MULTIPLY
4792 IF PRODUCT REQUIRES MORE THAN
4793 2 BYTES - CONVERT TO FLOATING-
4794
4795 POINT + USE FLOATING-POINT
4796 MULTIPLY
4797  # IN HL - MULTIPLICAND (PUT IN
4798                          BC AT 10ØB)
4799  # IN DE - MULTIPLIER
```

```
1085 3E 10          MVI  A,H'10' SET BIT CTR TO 16 DECIMAL (16 BITS TO MULTIPLY)
1087 29       X1087 DAD  H      MULTIPLY ACCUMULATED PRODUCT BY 2
1088 DA A0 10       JC   X10A0 OVERFLOW - ACCUMULATED PRODUCT TOO BIG FOR 2 BYTES - CONVERT TO FLOATING POINT
108B EB             XCHG } TEST NEXT BIT OF MULTIPLIER (HI ORDER TO LO ORDER)
108C 29             DAD  H } METHOD: DAD H SAME AS MULTIPLY BY 2; IF CARRY, HI BIT OF MULTIPLIER
108D EB             XCHG } WAS 1, MULTIPLY ACCUMULATED PRODUCT BY 2, THEN ADD MULTIPLICAND TO PRODUCT
108E D2 95 10       JNC  X1095 NO CARRY = MULTIPLIER BIT WAS 0
1091 09             DAD  B   ADD MULTIPLICAND TO ACCUMULATED PRODUCT IF HI BIT OF MULTIPLIER WAS A 1
1092 DA A0 10       JC   X10A0 IF CARRY OCCURS, ACCUMULATED PRODUCT IS TOO BIG FOR 2 BYTES - CONVERT TO FLOATING POINT
1095 3D       X1095 DCR  A  DECREMENT BIT CTR
1096 C2 87 10       JNZ  X1087 IF BIT CTR IS 00, 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       X109B 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 8000H = 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 0412
10B2 E1             POP  H PUT 1ST # IN HL
10B3 CD 36 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 0412
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 2D 1C       JM   X1C2D IF PRODUCT SHOULD BE MINUS, THEN 8000H = -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
10EA D2 EE 10       JNC  X10EE IF NO CARRY, DIVISOR IS LARGER THAN DIVIDEND - SUBTRACT NOT LEGITIMATE
10ED F1             POP  PSW ELIMINATE BITS OF DIVIDEND FROM STACK - SUBTRACT OK - RESULTS ALREADY IN HL
10EE 37             STC  SUBTRACT IS LEGITIMATE - SET CARRY TO SHIFT A 1 INTO ACCUMULATED QUOTIENT (IN DE)
10EF 3E E1    POP H MVI  A,H'E1' DUMMY RESTORE HL TO VALUE BEFORE SUBTRACT
```

INTEGER DIVIDE - \

(# IN DE) ÷ (# IN HL)

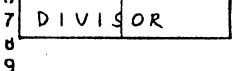
ARG1 ARG2



SHIFT LEFT 1 BIT AT 1DEF - 1DFA

CY IS 1 IF SUBTRACT OK

CY IS 0 IF SUBTRACT NOT OK



```
10EF  79        X10EF  MOV   A,E  } SHIFT 2 BYTES IN DE LEFT 1 PLACE
10F0  17               RAL        } - MULTIPLY QUOTIENT BY 2 (ALSO PUSH A BIT OF
10F1  5F               MOV   E,A  } (-DIVIDEND) INTO HL)
10F2  7A               MOV   A,D  }
10F3  17               RAL        }
10F4  57               MOV   D,A  }
10F5  7D               MOV   A,L  } SHIFT 2 BYTES IN HL LEFT 1 PLACE
10F6  17               RAL        }
10F7  6F               MOV   L,A  } - MULTIPLY NEGATED DIVIDEND BY 2
10F8  7C               MOV   A,H  }
10F9  17               RAL        }
10FA  67               MOV   H,A  }
10FB  F1               POP   PSW FETCH BIT-LOOP CTR FROM STACK
10FC  3D               DCR   A   DECREMENT BIT-LOOP CTR
10FD  C2 E5 10         JNZ   X10E5 CONTINUE LOOPING UNTIL ALL 17 BITS OF DIVISION COMPLETED
1E00  EB               XCHG  PUT QUOTIENT IN HL, PUT REMAINDER OF (ARG1/ARG2)*ARG2
1E01  C1               POP   D FETCH SIGN-OF-RESULT BYTE FROM STACK
1E02  D5               PUSH  D PUT SOMETHING ON STACK TO MATCH POP D AT 10A0 (THIS RESULT WILL NEVER FORCE A
                             BRANCH TO 10A8) - RESULT - REMAINDER IS PRESERVED IN DE
1E03  C3 98 10         JMP   X1098 COMBINE SIGN BIT WITH RESULT - STORE RESULT IN HOLDING AREA
----------------------------------------------------------------------------------------------
1E06  7C        X1E06  MOV   A,H PUT HI-ORDER BYTE OF 1ST # IN ACC } COMPUTE SIGN BIT OF
1E07  AA               XRA   D  XOR WITH HI-ORDER BYTE OF 2ND #   } INTEGER PRODUCT
1E08  47               MOV   B,A STORE RESULT IN B
1E09  CD 0D 1E         CALL  X1E0D TEST 1ST # IN HL, IF NEGATIVE, CONVERT TO POSITIVE USING 2'S COMPLEMENT
1E0C  EB               XCHG  SWAP #'S, 2ND # IN HL
1E0D  7C        X1E0D  MOV   A,H } TEST # IN HL
1E0E  B7        X1E0E  ORA   A   } IF # IS NEGATIVE, CONVERT TO POSITIVE USING 2'S COMPLEMENT
1E0F  F2 20 1C         JP    X1C20 } IF # IS POSITIVE, JMP, STORE RESULT IN HOLDING AREA
        ; FOR INTEGER MULTIPLY (OR DIVIDE), COMPUTE SIGN OF RESULT (XOR OF SIGN
        ; BITS), STORE IN B, CONVERT BOTH #'S TO POSITIVE INTEGERS
----------------------------------------------------------------------------------------------
1E12  AF        X1E12  XRA   A   CLEAR ACC
1E13  4F               MOV   C,A SAVE 00 IN C
1E14  95               SUB   L   SUBTRACT LO-BYTE OF INTEGER FROM 00
1E15  6F               MOV   L,A PUT RESULT BACK IN L
1E16  79               MOV   A,C PUT 00 IN ACC
1E17  9C               SBB   H   SUBTRACT HI-BYTE OF INTEGER FROM 00 WITH BORROW
1E18  67               MOV   H,A PUT RESULT BACK IN H
1E19  C3 20 1C         JMP   X1C20 STORE INTEGER VALUE IN HOLDING AREA - SET VALUE TYPE TO 02
        ; NEGATE INTEGER VALUE IN HL
        ; BY SUBTRACTING FROM 0000
----------------------------------------------------------------------------------------------
1E1C  2A 12 04  X1E1C  LHLD  X0412 LOAD HL WITH INTEGER VALUE FROM HOLDING AREA
1E1F  CD 12 1E         CALL  X1E12 NEGATE INTEGER VALUE, STORE BACK INTO HOLDING AREA
1E22  7C               MOV   A,H   } TEST IF INTEGER RESULT OF NEGATION IN 1E12 IS 8000
1E23  EE 80            XRI   H'80' }
1E25  B5               ORA   L
1E26  C0        <DONE  RNZ   RETURN IF RESULT ≠ 8000 (-32768)
1E27  EB               XCHG  PUT INTEGER # IN DE (LO ORDER 2 BYTES OF SNG-PREC #)
1E28  CD 83 1C         CALL  X1C83 SET VALUE TYPE TO SINGLE PRECISION
1E2B  AF               XRA   A   CLEAR ACC
1E2C  06 98     X1E2C  MVI   B,H'98' SET VALUE FOR EXPONENT TO 98 (= 2^23 -> 98H - 81H = 23 10)
1E2E  C3 FA 1A         JMP   X1AFA CONVERT # IN BADE TO SNG-PREC (CONSIDER SIGN BIT IN ACC & USE NORMALIZE ROUTINE)
        ; SPECIAL CASE -(8000) = 8000
        ; CANNOT STORE NEGATED (-32768) AS
        ;   32768 IN INTEGER FORMAT
----------------------------------------------------------------------------------------------
1E31  D5        X1E31  PUSH  D   SAVE ARG1 ON STACK
1E32  CD CC 10         CALL  X10CC PERFORM INTEGER DIVIDE - (# IN DE)\(# IN HL) - DE HAS MOD RESULT
1E35  AF               XRA   A   CLEAR ACC & CARRY BIT
1E36  82               ADD   D   } PUT REMAINDER TERM (ARG1 MOD ARG2) IN HL
1E37  1F               RAR       } RIGHT SHIFT ADJUSTS REMAINDER - COMPENSATES
1E38  67               MOV   H,A } FOR 17th SHIFT IN INTEGER DIVIDE
1E39  7B               MOV   A,E }
1E3A  1F               RAR       }
1E3B  6F               MOV   L,A }
1E3C  CD 30 1C         CALL  X1C30 SET VALUE TYPE (LOC 03A4) TO INTEGER TYPE (STORE 02)
1E3F  F1               POP   PSW FETCH ARG1 FROM STACK - SIGN OF RESULT SAME AS SIGN OF ARG1
1E40  C3 0E 1E         JMP   X1E0E NEGATE RESULT IF ARG1 IS NEGATIVE - STORE RESULT IN HOLDING AREA
        ; MOD
        ; ARG1 MOD ARG2
        ; (# IN DE)   (# IN HL)
        ; A MOD B = A-[B*A\B]
----------------------------------------------------------------------------------------------
1E43  21 1E 04         LXI   H,X041E LOAD HL WITH ADDR OF MSB(SIGN) BYTE OF # TO SUBTRACT
1E46  7E               MOV   A,M PUT SIGN BYTE IN ACC
        ; DOUBLE-PRECISION SUBTRACT
        ; (# IN 040E-0415) - (# IN 0418-041F
```

```
1E47  EE 80          XRI  H,80H 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   }
1E4F  C8        DONE 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
1E58  CA 88 18  DONE JZ   X1888 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 BYTES SAME AS FOR SNG-PREC # IN BCDE FOR 1873
1E73  2B             DCX  H } PUT SIGN + MSB BYTE OF 2ND # IN C
1E74  4E             MOV  C,M }
1E75  F1             POP  PSW FETCH EXPONENT DIFFERENCE FROM STACK
1E76  FE 39     X1E76 CPI  A'9' } TEST IF EXPONENT DIFFERENCE IS 57. OR GREATER
1E78  D0        DONE 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 18       CALL X1873 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 00 04       STA  X0400 } STORE THIS BYTE AT THE LOC (Ø4ØØ) 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-Ø415
1E96  D2 EA 1E  ALMOST DONE 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
1E9B  CA 9E 19  ERROR 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
```

DOUBLE-PRECISION ADD
(1ST #) ± (2ND #)
Ø4ØE...Ø415 Ø418...Ø41F
(MOVE 8 BYTES Ø418... → Ø4ØE)

```
1EAA  7E                 MOV  A,M }  COMPLEMENT LOC Ø41F
1EAB  2F                 CMA         (CANCELLED BY SAME SEQUENCE AT 1F31 - 1F33 IF CALL AT 1EAD IS EXECUTED)
1EAC  77                 MOV  M,A }
1EAD  DC 31 1F           CC   X1F31 IF CARRY SET (OVERFLOW = WRONG SIGN ON #) - CHANGE SIGN OF RESULT (STORED AT
1EB0  AF        X1EB0    XRA  A        CLEAR ACC (THIS LOOP ALLOWS 8 1-BYTE (8-BIT) LEFT SHIFTS AS PART OF      NORMALIZING
1EB1  47        X1EB1    MOV  B,A      SAVE SHIFT CTR IN B (COUNTS # OF BIT SHIFTS - USED TO ADJUST EXPONENT BYTE)
1EB2  3A 14 04           LDA  X0414 } PUT HI-ORDER BYTE OF # TO BE NORMALIZED IN ACC + TEST IT
1EB5  B7                 ORA  A     }
1EB6  C2 D9 1E           JNZ  X1ED9 IF HI-ORDER BYTE IS NOT ZERO, THEN FULL-BYTE SHIFT NOT APPLICABLE, JMP TO SINGLE BIT SHIFT ROUTINE
1EB9  21 0D 04           LXI  H,X040D LOAD HL WITH ADDR OF LOST-BIT BYTE OF RESULT (BYTE BELOW HOLDING AREA)
1EBC  0E 08              MVI  C,H'08' SET BYTE-LOOP CTR TO Ø8 (56 BITS OF MANTISSA + LOST BIT BYTE)
1EBE  56        X1EBE    MOV  D,M FETCH BYTE FROM MANTISSA INTO REG D
1EBF  77                 MOV  M,A STORE ACC INTO MANTISSA
1EC0  7A                 MOV  A,D MOVE BYTE FROM MANTISSA INTO ACC
1EC1  23                 INX  H  ADVANCE PTR TO MANTISSA
1EC2  0D                 DCR  C  DECREMENT BYTE-LOOP CTR
1EC3  C2 BE 1E           JNZ  X1EBE LOOP UNTIL ALL 8 BYTES HAVE BEEN SHIFTED
1EC6  78                 MOV  A,B PUT SHIFT CTR IN ACC FROM B
1EC7  06 08              SUI  H'08' SUBTRACT 8 (EACH BYTE SHIFT SAME AS 8 BIT SHIFTS, SUBTRACT 8 FROM EXPONENT FOR EACH SHIFT)
1EC9  FE C0              CPI  H'C0' TEST IF 8 BYTE SHIFTS HAVE BEEN DONE YET (64 BIT SHIFTS)
1ECB  C2 B1 1E           JNZ  X1EB1 JMP TO 1EB1 IF MORE SHIFTS TO DO, OTHERWISE ALL 8 BYTES WERE FOUND TO BE Ø. THEREFORE RESULT IS Ø
1ECE  C3 02 19   TOO SMALL JMP X1902 SET EXPONENT BYTE TO ZERO, RESULT IN HOLDING AREA
1ED1  05        X1ED1    DCR  B     DECREMENT SHIFT CTR (EACH SHIFT SAME AS MULTIPLY BY 2, SUBTRACT 1 FROM EXPONENT TO COMPENSATE)
1ED2  21 0D 04           LXI  H,X040D LOAD HL WITH ADDR OF LOST-BIT BYTE OF RESULT IN HOLDING AREA
1ED5  CD 78 1F           CALL X1F78 SHIFT 56 BIT MANTISSA + LOST-BIT BYTE LEFT 1 BIT
1ED8  B7                 ORA  A        TEST SHIFT CTR
1ED9  F2 D1 1E  X1ED9    JP   X1ED1 IF MSB OF # IS NOT A 1, THEN JMP TO 1ED1 TO DO A 1 BIT LEFT SHIFT
1EDC  78                 MOV  A,B   PUT SHIFT CTR IN ACC FROM B
1EDD  B7                 ORA  A     TEST SHIFT CTR
1EDE  CA EA 1E           JZ   X1EEA IF SHIFT CTR IS ZERO, PERFORM ROUNDOFF, RESULT STORED IN HOLDING AREA
1EE1  21 15 04           LXI  H,X0415 LOAD HL WITH ADDR OF EXPONENT OF RESULT
1EE4  86                 ADD  M     COMPUTE EXPONENT = EXPONENT - (# OF SHIFTS TO NORMALIZE)
1EE5  77                 MOV  M,A   PUT BYTE BACK INTO RESULT
1EE6  D2 02 19   DONE    JNC  X1902 IF NO CARRY, NORMALIZED # IS SO SMALL THAT IT IS SAME AS Ø
1EE9  C8         DONE    RZ         IF EXPONENT BYTE IS ZERO, NORMALIZED # SAME AS Ø, ADD IS DONE
1EEA  3A 0D 04  X1EEA    LDA  X040D LOAD ACC WITH LOST-BIT BYTE OF RESULT
1EED  B7        X1EED    ORA  A     TEST LOST BIT (MSB OF ACC)
1EEE  FC FC 1E           CM   X1EFC IF BIT SHIFTED OUT OF SMALLER MANTISSA WAS A 1, ROUND UP BY ADDING 1 TO THE # AT Ø40E ... Ø414
1EF1  21 16 04           LXI  H,X0416 } PUT BYTE WITH SIGN OF LARGER # IN ACC
1EF4  7E                 MOV  A,M     }    (or # IN HOLDING AREA)
1EF5  E6 80              ANI  H'80' ELIMINATE ALL BUT THE SIGN BIT
1EF7  2B                 DCX  H }  BACK-UP PTR TO MSB OF MANTISSA Ø414
1EF8  2B                 DCX  H }
1EF9  AE                 XRA  M COMBINE SIGN BIT WITH HI 7 BITS OF MANTISSA OF RESULT (NOTE: LEADING 1 LOST)
1EFA  77                 MOV  M,A PUT BYTE BACK INTO RESULT IN HOLDING AREA
1EFB  C9                 RET  DONE
-----------------------------------------------------------------------------------------------------------
1EFC  21 0E 04  X1EFC    LXI  H,X040E LOAD HL WITH ADDR OF LO-ORDER BYTE OF 56 BIT MANTISSA        ADD 1 TO MANTISSA AT Ø40E...Ø414
1EFF  06 07              MVI  B,H'07' SET LOOP CTR TO 7 (7 BYTES IN MEMORY ARE THE MANTISSA)     (56 BIT MANTISSA, EXPONENT OF
1F01  34        X1F01    INR  M ADD 1 TO BYTE IN MANTISSA                                         RESULT SAME AS EXPONENT OF
1F02  C0         DONE    RNZ  IF NOT ZERO, DONE      IF ZERO, CARRY 1 INTO NEXT BYTE              LARGER #, IF OVERFLOW, ADD 1
1F03  23                 INX  H ADVANCE PTR TO NEXT BYTE IN MANTISSA                              TO EXPONENT, FORCE MANTISSA
1F04  05                 DCR  B DECREMENT LOOP CTR                                                TO 8Ø ØØ... ØØ
1F05  C2 01 1F           JNZ  X1F01 LOOP UNTIL ALL 7 BYTES IN MANTISSA HAVE BEEN INCREMENTED
1F08  34                 INR  M ADJUST EXPONENT BYTE (ADD 1, MULTIPLY # BY 2)
1F09  CA 3E 19   ERROR   JZ   X193E IF ZERO, EXPONENT BYTE OVERFLOW, PRINT ERROR MESSAGE "OVERFLOW"
1F0C  2B                 DCX  H BACK-UP PTR TO HI-ORDER BYTE IN MANTISSA (Ø414)
1F0D  36 80              MVI  M,H'80' DIVIDE ENTIRE MANTISSA BY 2 (PUTS OVERFLOW BACK IN MANTISSA)
1F0F  C9                 RET  DONE WITH ROUND-UP ROUTINE
-----------------------------------------------------------------------------------------------------------
1F10  11 3B 04  X1F10    LXI  D,X043B LOAD DE WITH ADDR OF LSB OF DIVIDEND                        LOAD DE + HL WITH ADDR'S OF 2 #'S
1F13  21 1A 04           LXI  H,X041A LOAD HL WITH ADDR OF LSB OF DIVISOR                         TO BE ADDED OR SUBTRACTED (USED
                                                                                                  BY ... 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  7E          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 58 1F          JC    X1F58 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)
```

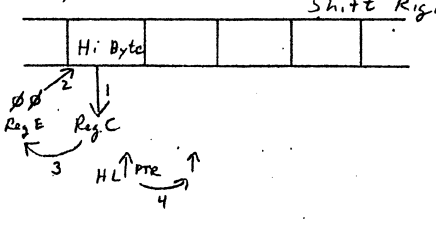
ADD OR SUBTRACT 2 SEVEN-BYTE #'S IN MEMORY

LARGER # } FROM IBAL
CONSTANT SMALLER #

COMPUTE Ø - MANTISSA
INVERT SIGN BIT

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

Shift Right



```
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 Ø4ØE-Ø414 TO Ø438-Ø441, CLEAR HOLDING AREA - USED
1F8B  71                MOV   M,C CLEAR LOC Ø44ØD (LOST-BIT BYTE OF ACCUMULATED PRODUCT)
1F8C  13                INX   D ADVANCE PTR TO MULTIPLIER  TO LOC Ø43B (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 (Ø418-Ø41E) TO ACCUMULATED PRODUCT (Ø4ØE-Ø414) - 7 BYTE MANTISSA
1F9E  CC 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, STORE RESULT IN HOLDING AREA
1FAF  21 14 04    X1FAF LXI   H,X0414 LOAD HL WITH ADDR OF MSB OF ACCUMULATED PRODUCT IN HOLDING AREA
1FB2  CC 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  CC 66 13          CALL  X1366 MOVE CONSTANT INTO TEMP STORAGE AREA

1FC9  3A 1F 04          LDA   X041F  TEST DIVISOR STORED IN TEMP STORAGE AREA Ø418....
1FCC  B7                ORA   A
1FCD  CA AC 04          JZ    X04AC  IF DIVISOR IS ZERO, PRINT ERROR MESSAGE "DIVISION BY ZERO"
1FD0  CD 9E 1A          CALL  X1A9E COMPUTE EXPONENT OF RESULT, RESTORE LEADING 1'S, COMPUTE XNOR OF SIGN BITS
1FD3  34                INR   M  ADD 2 TO EXPONENT OF RESULT (LOC Ø415)
1FD4  34                INR   M
1FD5  CC 18 20          CALL  X2018 MOVE DIVIDEND FROM Ø4ØE-Ø414 TO Ø438-Ø441, CLEAR HOLDING AREA - USED TO ACCUMULATE QUOTIENT
1FD8  21 42 04          LXI   H,X0442 LOAD HL WITH ADDR OF EXPONENT BYTE OF DIVIDEND
1FDB  71                MOV   M,C CLEAR LOC Ø442 (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 20D9-20DB
```

DOUBLE-PRECISION FLOATING-POINT CONSTANT = 10 DECIMAL

SINGLE-PRECISION FLOATING-POINT CONSTANT = 10 DECIMAL

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

DOUBLE-PRECISION MULTIPLY

(# IN Ø4ØE-Ø415) X (# IN Ø418-Ø41F)

MULTIPLIER TO ACCUMULATE PRODUCT

STORE RESULT IN HOLDING AREA

DOUBLE PRECISION DIVIDE

(# IN Ø4ØE-Ø415) ÷ (# IN Ø418-Ø41F)

```
                                                    (Ø43Ø-Ø44Ø) (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 ROUNDOFF TEST AT 5172 IEEE-IEEE
1FF7  FA EØ 1E          JM    X1EE0 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 H , 5177 OTHERWISE SHIFT A Ø INTO QUOTIENT
2003  21 3Ø 04          LXI   H,X043Ø 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
2009  78                MOV   A,B } TEST REG B - IF ENTIRE QUOTIENT IS ZERO, ADJUST EXPONENT BY 1      5180 BYTE
200A  B7                ORA   A   } (see 18BC-1896 - SNG-PREC DIVIDE)                 7     5181
200B  C2 ØD 1F          JNZ   X1F0D 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 ØD 1F          JNZ   X1F0D IF EXPONENT BYTE IS NOT ZERO, JMP TO 1F0D 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       B#      5198
202C  C9                RET   DONE                                             I       5199
------------------------------------------------------------------------------------------------------------
2020  CD 91 18    X2020 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 1Ø 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 1ØX # IN HOLDING AREA) A) 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 CONVERT STRING TO NUMERIC
2045  F5                PUSH  PSW  PUT CHAR & FLAGS ON STACK                                 5214 VALUE,
2046  CA 4F 20          JZ    X204F JMP IF CHAR IS '-'                           JO    5215
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 FF 00          LXI  B,X00FF PUT -1 IN C, PUT ØØ IN B
2053  60                MOV  H,B } INITIALIZE HL TO ØØØØ - ACCUMULATED VALUE STARTS AS
2054  68                MOV  L,B }                 AN INTEGER VALUE ØØØØ
2055  CD 20 1C          CALL X1C20 INITIALIZE VALUE HOLDING AREA TO ØØØØ + INTEGER TYPE      M-
2058  EB                XCHG PUT PTR TO CHAR STRING IN HL, PUT ØØØØ IN DE
2059  D7       ->X2059  RST  2   SCAN TO NEXT CHAR (SIGN CHAR, IF ANY, IS ON STACK - see 2045)  H
205A  DA E8 20 <-DIGIT- JC   X20E8 JMP IF CHAR IS A DIGIT-                                    Z
205D  FE 2E    '.'      CPI  A'.' } TEST CHAR, JMP IF '.'                                      .
205F  CA AF 20 <-       JZ   X20AF }                                                           J/
2062  FE 45    'E'      CPI  A'E' } TEST CHAR, JMP IF 'E', CONVERT VALUE IN HOLDING AREA TO SNG-PREC, CONTINUE TO PROCESS EXPONENT
2064  CA 6D 20          JZ   X206D }                                                           J
2067  FE 44             CPI  A'D' } TEST CHAR, JMP IF NOT 'D'                                  D
2069  C2 91 20 NOT 'D'  JNZ  X2091 }                                                           B
206C  B7       'D'      ORA  A  RESET THE ZERO FLAG (CHAR IN ACC IS 'D')                        7
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
2070  E5                PUSH H        } PUT ADDR 2086 ON STACK FOR IMPLIED JMP
2071  21 86 20          LXI  H,X2086  }                                                        !
2074  E3                XTHL          } LEAVE PTR TO CHAR STRING IN HL
2075  D7                RST  2  SCAN TO NEXT NON-SPACE CHAR IN STRING                          H
2076  15                DCR  D  DECREMENT D TO -1
2077  FE AF    '-'EXP   CPI  H'AF' } JMP TO 2086 WITH D = -1 IF CHAR IS MINUS SIGN OR          /
2079  C8       <-       RZ         } TOKEN FOR A MINUS SIGN                                    H
207A  FE 2D    '-'EXP   CPI  A'-'  }                                                           -
207C  C8       <-       RZ         }                                                           H
207D  14                INR  D  INCREMENT D TO ØØ
207F  FE AE    '+'EXP   CPI  H'AE' } JMP TO 2086 WITH D = ØØ IF CHAR IS PLUS SIGN OR           .
2080  C8       <-       RZ         } TOKEN FOR A PLUS SIGN                                     H
2081  FE 2B    '+'EXP   CPI  A'+'  }                                                           +
2083  C8       <-       RZ         }                                                           H
2084  2B                DCX  H  IF CHAR IS NOT A SIGN CHAR OR SIGN TOKEN, BACK-UP SCAN PTR     +
2085  F1                POP  PSW REMOVE IMPLIED JMP ADDR 2086 FROM STACK
2086  D7       ->X2086  RST  2  SCAN TO NEXT NON-SPACE CHAR                                    W
2087  DA 57 21 <-DIGIT- JC   X2157 JMP IF CHAR IS A DIGIT, IF NOT A DIGIT, THEN IT MUST BE A DELIMITER ZW!
208A  14                INR  D INCREMENT SIGN OF EXPONENT BYTE (SETS THE FLAGS CONVENIENTLY)
208B  C2 91 20 <-       JNZ  X2091 IF EXPONENT IS POSITIVE, JMP TO 2091                        B
208E  AF                XRA  A CLEAR ACC                           } FORM 2'S COMPLEMENT OF    /
208F  93                SUB  E SUBTRACT EXPONENT MAGNITUDE         } MAGNITUDE (SINCE MINUS EXPONENT)
2090  5F                MOV  E,A PUT # BACK INTO REG E                                         -
2091  E5       ->X2091  PUSH H SAVE PTR TO CHAR STRING ON STACK
2092  78                MOV  A,E } SUBTRACT #-OF-DIGITS-ON-RIGHT-OF-DECIMAL-PT FROM EXPONENT MAGNITUDE
2093  90                SUB  B   } (MANTISSA IS AN INTEGER AT THIS POINT
2094  F4 C9 20   X2094  CP   X20C9 IF EXPONENT IS POSITIVE, MULTIPLY MANTISSA BY 1Ø, DECREMENT ACC BY 1
2097  FC 07 20          CM   X2007 IF EXPONENT IS NEGATIVE, DIVIDE MANTISSA BY 1Ø, INCREMENT ACC BY 1  H
209A  C2 94 20          JNZ  X2094 LOOP UNTIL EXPONENT MAGNITUDE REDUCED TO Ø                  B
209D  E1                POP  H  FETCH PTR TO CHAR STRING FROM STACK
209E  F1                POP  PSW FETCH SIGN CHAR + FLAGS FROM STACK
209F  E5                PUSH H SAVE PTR TO CHAR STRING ON STACK
20A0  CC 0C 1B          CZ   X1B0C ZERO FLAG SET IF ENTIRE # IS MINUS (see 2043), NEGATE VALUE IN HOLDING AREA
20A3  E1                POP  H FETCH PTR TO CHAR STRING FROM STACK
20A4  F7                RST  6 TEST FOR VALUE TYPE IN HOLDING AREA
20A5  E8       <-DONE   RPE  CONVERSION FINISHED IF INTEGER OR DBL-PREC VALUE
20A6  E5                PUSH H SAVE PTR TO CHAR STRING ON STACK
20A7  21 1C 1A          LXI  H,X1A1C } PUT PSEUDO ERROR MESSAGE ADDR ON STACK, IF SNG-PREC # IS NOT
20AA  E5                PUSH H       } THE SPECIAL CASE, RET TO 1A1C WHICH POPS H (PTR TO STRING IN HL)
20AB  CD 16 1C          CALL X1C36 IF SNG-PREC VALUE IS SPECIAL CASE, CONVERT TO INTEGER FORMAT   M6
20AE  C9       <-DONE   RET  PUT PTR TO CHAR STRING BACK IN HL BEFORE EXITING THIS ROUTINE     I
20AF  F7       X20AF    RST  6 TEST VALUE TYPE OF ACCUMULATED VALUE IN HOLDING AREA
20B0  0C                INR  C CHANGE C TO ØØ - INDICATES NEXT GROUP OF DIGITS HAVE FRACTIONAL VALUE  B
20B1  C2 91 20          JNZ  X2091 IF C IS NOT Ø, SECOND DECIMAL-PT FOUND - END OF MAGNITUDE EVALUATION
20B4  DC BA 20          CC   X20BA CONVERT ACCUMULATED VALUE TO SINGLE PRECISION
20B7  C3 59 20          JMP  2059 CONTINUE PROCESSING NEXT CHAR
```

SPECIAL CASE: -32768 CAN BE STORED AS AN INTEGER-FORMAT VALUE. TEST IF SNG-PREC SHOULD BE AN INTEGER

DECIMAL-PT FOUND IN MANTISSA

```
2087  C3 5 20            JMP  X2059                                                                         CY
2088  E5         X208A    PUSH H                                                                             
2089  D5                  PUSH D } SAVE ALL REGISTERS ON THE STACK                                          U     CONDITIONAL CONVERSION TO
208C  C5                  PUSH B                                                                            E     SNG- OR DBL-PRECISION (# IN
208D  F5                  PUSH PSW SAVE FLAGS ON STACK (ZERO FLAG SET- SNG PREC, ZERO FLAG RESET-                  DBL PREC)  HOLDING AREA)
208E  CC 45 1C            CZ   X1C45 CONVERT VALUE IN HOLDING AREA TO SINGLE-PRECISION, IF ZERO FLAG      E       SET
20C1  F1                  POP  PSW FETCH FLAGS FROM STACK
20C2  C4 6F 1C            CNZ  X1C6F CONVERT VALUE IN HOLDING AREA TO DOUBLE-PRECISION, IF ZERO FLAG      O       RESET
20C5  C1                  POP  B } RESTORE REGISTERS TO ORIGINAL VALUES FROM STACK                        A
20C6  D1                  POP  D                                                                          G
20C7  E1                  POP  H
20C8  C9                  RET    END OF CONDITIONAL CONVERSION                                            I
20C9  C8         X20C9    RZ     RETURN IF EXPONENT CTR IS Ø (SINCE THIS ROUTINE IS CALLED BY A POSITIVE CONDITION  H   (See 2094
20CA  F5         X20CA    PUSH PSW SAVE EXPONENT CTR ON STACK
20CB  F7                  RST  6 TEST VALUE TYPE OF ACCUMULATED VALUE STORED IN HOLDING AREA
20CC  F5                  PUSH PSW SAVE FLAG STATUS ON STACK                                                      MULTIPLY ACCUMULATED VALUE
20CD  E4 04 1A            CPO  X1A04 MULTIPLY SINGLE-PREC # IN HOLDING AREA BY 1Ø DECIMAL                 T       BY 1Ø DECIMAL
2000  F1                  POP  PSW FETCH FLAG STATUS FROM STACK
2001  EC 20 20            CPE  X2020 MULTIPLY DOUBLE-PREC # IN HOLDING AREA BY 1Ø DECIMAL                 -
2004  F1                  POP  PSW FETCH EXPONENT CTR FROM STACK
2005  3D         X2005    DCR  A DECREMENT EXPONENT CTR (EACH POSITIVE 1 IN EXPONENT SAME AS =                    MULTIPLY BY 1Ø
2006  C9                  RET  DONE                                                                       I
2007  D5         X2007    PUSH D } SAVE REGISTERS ON STACK                                                U
2008  E5                  PUSH H
2009  F5                  PUSH PSW SAVE EXPONENT CTR ON STACK
200A  F7                  RST  6 TEST VALUE TYPE OF ACCUMULATED VALUE STORED IN HOLDING AREA
200B  F5                  PUSH PSW SAVE FLAG STATUS ON STACK                                                      MULTIPLY ACCUMULATED VALUE
200C  E4 23 1A            CPO  X1A23 DIVIDE SINGLE-PRECISION # IN HOLDING AREA BY 1Ø DECIMAL              #       BY .1 DECIMAL
200F  F1                  POP  PSW FETCH FLAG STATUS FROM STACK
20E0  EC C0 1F            CPE  X1FC0 DIVIDE DOUBLE-PRECISION # IN HOLDING AREA BY 1Ø DECIMAL              @
20E3  F1                  POP  PSW
20E4  E1                  POP  H } RESTORE REGISTERS TO ORIGINAL VALUES FROM STACK
20E5  D1                  POP  D                                                                          Q
20E6  3C                  INR  A INCREMENT EXPONENT CTR (EACH NEGATIVE 1 IN EXPONENT SAME AS <                    DIVIDE BY 1Ø)
20E7  C9                  RET  DONE                                                                       I
20E8  D5         X20E8    PUSH D SAVE EXPONENT SIGN (D) + EXPONENT MAGNITUDE (E) ON STACK (BOTH ARE       U       Ø HERE)
20E9  78                  MOV  A,B } ENTRY TO 20E8 WITH CARRY SET (see 205A), B IS Ø, C IS -1 ORIGINALLY,         WHEN DECIMAL PT IS FOUND, C BECOMES ØØ
20EA  89                  ADC  C     EACH DIGIT ON RIGHT OF DECIMAL-PT ADDS 1 TO B
20EB  47                  MOV  B,A                                                                        G
20EC  C5                  PUSH B SAVE B (# OF DIGITS TO RIGHT OF DECIMAL-PT) + C (DECIMAL-PT INDICATOR) E ON      STACK
20ED  E5                  PUSH H SAVE PTR TO CHAR STRING ON STACK
20EE  7E                  MOV  A,M FETCH CHAR FROM STRING INTO ACC
20EF  D6 30               SUI  A'0' } CONVERT ASCII CHAR TO A DIGIT-WEIGHT (INTEGER FORMAT)              V0
20F1  F5                  PUSH PSW } SAVE DIGIT-WEIGHT ON STACK
20F2  F7                  RST  6 TEST VALUE TYPE OF # IN HOLDING AREA
20F3  F2 1D 21  FLT'G PT  JP   X211D JMP IF FLOATING-PT TYPE OF VALUE (SNG- OR DBL-PREC)                  !
20F6  2A 12 04            LHLD X0412 LOAD HL WITH ACCUMULATED INTEGER VALUE FROM HOLDING AREA             *
20F9  11 C0 0C            LXI  D,X0CCD LOAD DE WITH 3277 DECIMAL (MAX POSITIVE INTEGER # IS 32767.        M
20FC  E7                  RST  4 TEST IF ACCUMULATED INTEGER VALUE IS TOO LARGE TO ACCOMODATE                      ANOTHER DIGIT (VALUE x 1Ø + DIGIT-WEIGHT)
20FD  D2 19 21  CONVERT TO SNG-PREC  JNC  X2119 IF HL ≥ DE, ACCUMULATED VALUE TOO LARGE FOR INTEGER FORMAT !,           CONVERT TO SINGLE-PRECISION - JMP
2100  54         X2100    MOV  D,H } STORE # IN HL IN DE } MULTIPLY ACCUMULATED VALUE                      T
2101  5D                  MOV  E,L }                       IN HL BY 1Ø DECIMAL                            )
2102  29                  DAD  H 2X                                                                       )
2103  29                  DAD  H 4X                                                                       )
2104  19                  DAD  D +1 = 5X                                                                  )
2105  29                  DAD  H x2 = 10X
2106  F1                  POP  PSW FETCH DIGIT-WEIGHT FROM STACK                                          O
2107  4F                  MOV  C,A } ADD DIGIT-WEIGHT TO ACCUMULATED VALUE
2108  09                  DAD  B
2109  7C                  MOV  A,H PUT HI-BYTE OF ACCUMULATED VALUE IN ACC
```

```
210A B7          ORA  A TEST HI-BYTE,                                    7    5340
2108 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    A    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)       O    5345
2114 C3 59 20    JMP  X2059 CONTINUE PROCESSING NEXT CHAR IN STRING  NEXT CHAR      CY   5346
2117 79      X2117 MOV  A,C } PUT DIGIT-WEIGHT ON STACK, CONVERT ACCUMULATED VALUE TO  5347
211A 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 X2110 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 1D    CALL X1DA1 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)          G!   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    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 1D    CALL X1D91 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    JMP  X2111 RESTORE PTRS FROM STACK, CONTINUE PROCESSING NEXT CHAR IN C !  5367 STRING
214C CD 36 18 X214C CALL X1836 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  K    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 7B      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  } MULTIPLY ACCUMULATED EXPONENT VALUE BY 10. DECIMAL          5376
215E 07          RLC  }                                                             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' }                                                   V0   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
2169 C3 86 20    JMP  X2086 LOOP TO ACCUMULATE MORE EXPONENT DIGITS            C    5384
216B E5      X2168 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 20 1C X2173 CALL X1C20 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 22 22    CALL X2222 CLEAR 03CD, PUT SPACE AT FIRST LOC IN CHAR BUFR     M "  5391
217A B6          ORA  M PUT '␣' IN ACC                                         6    5392
217B CD 9E 21    CALL X219E CONVERT # IN HOLDING AREA (0412 & 0413) TO ASCII IN CHARM !  5393 BUFR
217E C3 40 12    JMP  X1240 PRINT CHAR STRING (CONVERTED #)                    CQ   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
```

CONVERT & PRINT LINE # IN HL

```
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
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 1B0C, 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 A4 03              LDA   X03A4 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 0B 22  SEPARATE RETURN AT 220B  JPE  X220B JMP IF VALUE IS SNG- OR DBL-PREC FLOATING-PT VALUE - CONVERT VALUE TO CHAR
21B4 01 00 00  UNFORMATTED FLTING-PT CONVERSION  LXI  B,X0000 SET B&C SO NO COMMA'S or DECIMAL PT IN CONVERTED #
21B7 CD FA 24              CALL  X24FA CONVERT # TO ASCII CHAR STRING STARTING AT 0422
21BA 21 21 04      X21BA   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 2# 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 2##0
21D1 71            X21D1   MOV   M,C PUT '␣' or '*' in memory - '*' IF DOING '*' FILL, '␣' OTHERWISE
21D2 D7            END OF # STRING  RST  2  SCAN TO NXT NON-SPACE CHAR IN CONVERTED # STRING - SET FLAGS, NON-SPACE CHAR IN ACC
21D3 CA EF 21              JZ    X21EF JMP IF END OF CHAR STRING HAS BEEN FOUND (CHAR = NULL)
21D6 FE 45         'E'     CPI   A'E' }
21D8 CA EF 21              JZ    X21EF } JMP IF CHAR IS 'E' } NO MORE LEADING '0' SUPPRESSION
21DB FE 44         'D'     CPI   A'D' } JMP IF CHAR IS 'D' } OR LEADING '*' FILL
21DD CA EF 21              JZ    X21EE }
21E0 FE 30         '0'     CPI   A'0' } IF CHAR IS A '0' (LEADING 0) REPLACE WITH A SPACE OR '*0' - WHICHEVER IS REQUIRED
21E2 CA D1 21              JZ    X21D1 }     (LEADING '0' SUPPRESSION)
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 3E 30                 MVI   M,A'0' PUT A SINGLE '0' CHAR IN BUFR (BEFORE END OF BUFR, 'E', 'D', or '.')
21F2 78            X21F2   MOV   A,E  } PUT BIT-FLAG BYTE IN ACC + TEST THE 1# 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 78            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

5400 (IN 1B25)
5401 M%
5402 21AC
5403
5404 6-
5405 E
5406
5407 IS A POSITIVE #
5408
5409 A
5410 4
5411
5412 LEADING ZERO
5413 :M
5414 W
5415                0421  '␣' or sign char
5416 :$             0422  ASCII DIGIT HI ORDER
5417 Z **           0423
5418
5419
5420 CHAR STRING
5421                0XX ASCII DIGIT LO ORDER
5422 M $                  0
5423 !!             INTEGER # CONVERT - UNFORMATTED
5424 F
5425                SUPPRESS LEADING ZERO'S
5426 :M             INSERT SIGN CHAR
5427 -              LEADING '*' FILL '$' INSERT
5428
5429 JQ!
5430               LEADING '**' FILL
5431
5432 *
5433 BQ! *
5434 A FOR 22##
5435
5436 NON-SPACE CHAR IN ACC
5437 !
5438 E
5439 J !
5440 D
5441 J !
5442
5443 JQ!
5444 ,
5445 JQ!
5446 .
5447 9 !
5448 +              END OF STRING '0' 'E' '.'
5449 60 .)
5450
5451
5452 J !
5453 +
5454 6$
5455
5456
5457 @
5458 +
5459
```

```
2201  C9                    RET                                                         I
2202  32 CD 03    X2202     STA   X03CD STA BIT-FLAG BYTE AT FLAG LOC Ø3CD              2M
2205  21 21 04              LXI   H,X0421 } PUT A SPACE AT LOC Ø421 (2 LOC ABOVE OUTPUT  !!   - INITIAL CHAR FOR A SUPPRESSED '+' SIGN
2208  36 20                 MVI   M,A' '  }                                             6
220A  C9                    RET                        BUFFER AREA)                     I
220B  E5          X2208     PUSH  H     SAVE PTR. TO CHAR BUFR ON STACK (PT TO LOC Ø422)          FLOATING-PT # CONVERT
220C  DE 00  CARRY SET FOR  SBI   H'00' SET FLAGS - (CLEARS CARRY BIT)  IN ACC DBL-PREC 8  SNG-PREC 3   ^   UNFORMATTED OUTPUT
220E  17     SNG-PREC       RAL   MULTIPLY BY 2   IN ACC (SNG-PREC 6) (DBL-PREC 16)
220F  57                    MOV   D,A  STORE RESULT IN D                                H
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 1Ø^5 ≤ # < 1Ø^6) or (DBL-PREC 1Ø^15 ≤ # < 1Ø^16)  MX#   ACC HAS EXPONENT ADJUSTMENT  X1Ø'S -> - ACC
2214  01 00 03              LXI   B,X0300 LOAD DECIMAL-PT POSITION CTR WITH Ø3 (B), SET COMMA POSITION CTR TO Ø (NO COMMA'S)   ÷1Ø'S -> + ACC
2217  82     SNG #<1Ø^-5    ADD   D  } TEST DECIMAL EXPONENT-ADJUST VALUE (OBTAINED FROM 2308)
2218  FA 24 22  DBL #≤1Ø^-3 JM    X2224} JMP IF # ORIGINAL VALUE (SNG-PREC # < 1Ø^-5) or (DBL-PREC # < 1Ø^-3)   $"
221B  14     ACC IS NEG #   INR   D  } TEST DECIMAL EXPONENT-ADJUST VALUE AGAIN
221C  BA     SNG #≥1Ø^6     CMP   D
221D  D2 24 22  DBL #≥1Ø^16 JNC   X2224} JMP IF # ORIGINAL VALUE (SNG-PREC # ≥ 1Ø^6) or (DBL-PREC # ≥ 1Ø^16)  R$"
2220  3C     ACC IS POS #   INR   A  ADD 1 TO COMPENSATE FOR DCR B AT START OF LOOP AT 2451}   <
2221  47                    MOV   B,A PUT NEW VALUE FOR DECIMAL-PT POSITION CTR IN B   }  G
2222  3E 02                 MVI   A,H'02' LOAD ACC WITH Ø2 SO ZERO FLAG WILL BE SET BY NXT INSTRUCTION TO >FORCE NON-EXPONENT OUTPUT FORMAT
2224  D6 02       X2224     SUI   H'02' SET OR CLEAR ZERO FLAG - IF RESET, USE EXPONENTIAL FORMAT   V
2226  E1                    POP   H FETCH PTR TO CHAR BUFR FROM STACK
2227  F5   ONL 'Ø' IN BUFR  PUSH  PSW SAVE FLAG STATUS ON STACK
2228  CD 51 24  FROM 219F   CALL  X2451 INSERT DP IN CHAR BUFR IF NECESSARY AT LOC Ø422 (KILLS LEADING ZERO PUT IN AT 219F)  MQ3
222B  36 30                 MVI   M,A'0'                                                60
222D  CC 5B 19              CZ    X195B                                                 L(
2230  CD 65 24              CALL  X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT  - IF INTEGER, DP LAST CHAR IN BUFR
2233  2B          X2233     DCX   H BACK-UP PTR TO CHAR BUFR                            +
2234  7E                    MOV   A,M FETCH CHAR FROM CONVERTED #
2235  FE 30                 CPI   A'0' TEST CHAR                                        0
2237  CA 33 22              JZ    X2233 JMP IF CHAR IS 'Ø' - LOOP UNTIL ALL TRAILING 'Ø' CHAR'S TO RIGHT OF DP  J3"  HAVE BEEN REMOVED
223A  FE 2E                 CPI   A'.' TEST CHAR                                        .
223C  C4 5B 1B              CNZ   X1B5B IF CHAR IS NOT '.', LEAVE CHAR IN BUFR AS PART OF RESULT BY DOING  D(AN  INX H
223F  F1                    POP   PSW FETCH FLAG STATUS FROM STACK
2240  CA 63 22  NO EXPONENT TO CONVERT  JZ  X2263 JMP IF (SNG PREC 1Ø^-5 ≤ # < 1Ø^6) or (DBL PREC 1Ø^-3 ≤ # < 1Ø^16) (EXPONENT FORMAT J NOT BEING USED)
2243  F5          X2243     PUSH  PSW SAVE FLAG STATUS ON STACK + EXPONENT VALUE          ↓ CONVERT + PRINT
2244  F7                    RST   6  TEST TYPE OF VALUE IN HOLDING (# TO BE CONVERTED) RNC - DBL-PREC  RC - SNG-PREC   EXPONENT OF #
2245  3E 22                 MVI   A,A'"' LOAD ACC WITH 22 (1/2 OF 'D')                  >"
2247  8F                    ADC   A  COMPUTE A CHAR, IF # IS DBL-PREC, CHAR IS 'D', IF # IS SNG-PREC, CHAR IS 'E'
2248  77                    MOV   M,A  PUT CHAR FOR EXPONENT IN CHAR BUFR
2249  23                    INX   H  ADVANCE PTR TO CHAR BUFR                           #
224A  F1                    POP   PSW FETCH FLAG STATUS FROM STACK + EXPONENT VALUE
224B  36 2B                 MVI   M,A'+' PUT A '+' IN CHAR BUFR FOR EXPONENT            6+
224D  F2 54 22              JP    X2254 IF EXPONENT IS POSITIVE, JMP TO 2254 TO CONVERT EXPONENT MAGNITUDE  T"
2250  36 2D                 MVI   M,A'-' PUT A '-' IN CHAR BUFR FOR EXPONENT            6-
2252  2F                    CMA   } FORM 2'S COMPLEMENT OF EXPONENT MAGNITUDE           /
2253  3C                    INR   A   (POSITIVE # IN ACC)                               <
2254  06 2F       X2254     MVI   B,A'/' SET CHAR INITIALLY TO '/' (1 LESS THAN 'Ø')    /
2256  04          X2256     INR   B  ADVANCE CHAR COUNT BY 1
2257  D6 0A                 SUI   H'0A' SUBTRACT 1Ø FROM EXPONENT MAGNITUDE             V
2259  D2 56 22              JNC   X2256 LOOP UNTIL EXPONENT MAGNITUDE IS LESS THAN 1Ø  RV"
225C  C6 3A                 ACI   A':' CONVERT REMAINDER IN ACC TO A CHAR ('Ø'-'9') - ADD 1 MORE THAN '9' TO # IN ACC   F:
225E  23                    INX   H  ADVANCE PTR TO CHAR BUFR                           #
225F  70                    MOV   M,B  PUT 1ST CHAR OF EXPONENT MAGNITUDE IN CHAR BUFR
2260  23                    INX   H  ADVANCE PTR TO CHAR BUFR                           #
2261  77                    MOV   M,A PUT 2ND CHAR OF EXPONENT MAGNITUDE IN CHAR BUFR
2262  23          X2262     INX   H  ADVANCE PTR TO CHAR BUFR                           #   ↓ TERMINATE CHAR BUFFER
2263  36 00       X2263     MVI   M,H'00' PUT A NULL IN CHAR BUFR TO TERMINATE CONVERTED-# STRING   6
2265  EB                    XCHG  PUT ADDR OF NULL OF CONVERTED # STRING IN DE (USED BY 23D3)
2266  21 21 04              LXI   H,X0421 LOAD HL WITH ADDR OF 1ST CHAR OF CONVERTED #  !!
                            RET
```

```
2259  C9                                                                      I    5520
226A  23             X226A  INX   H ADVANCE PTR TO LOC Ø423 (LOC AFTER 1ST LEADING 'Ø')             Ø    5521
226B  C5                    PUSH  B PUT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP (B) & # OF NUMERIC FIELD POSITIONS TO RIGHT  E  5522 OF DP (INCLUDING DP) (C) ON STACK (see 1671-1673)
226C  FE 04                 CPI   H'04' TEST VALUE TYPE CODE OF # TO CONVERT (PUT IN ACC AT 21A6)        5523
226E  7A                    MOV   A,D  PUT BIT FLAG BYTE IN ACC                                          5524
226F  D2 E0 22  <-          JNC   X22E0 JMP IF VALUE TO CONVERT IS FLOATING-PT (SNG-OR DBL-PREC)    R ''  5525
2272  1F                    RAR   TEST Ø1 BIT OF BIT-FLAG BYTE IN ACC (see p. 88)                        5526  INTEGER VALUE CONVERT-FORMATTED OUTPUT
2273  DA 78 23  <-          JC    X2378 JMP IF INTEGER VALUE TO BE PRINTED IN EXPONENTIAL FORMAT    Z Ø  5527
2276  01 03 06              LXI   B,X0603 LOAD B WITH A DIGIT COUNT FOR DP POSITION (5 DIGITS + 1=COUNT), LOAD C WITH  5528 DIGIT COUNT FOR COMMA POSITION
2279  CD 49 24              CALL  X2449 TEST BIT-FLAG FOR COMMA FORMAT IN CONVERTED # - CLEAR C IF NO COMMA'S  HI$  5529 IN FORMAT
227C  D1                    POP   D PUT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP IN D             0    5530      # IN D
227D  7A                    MOV   A,D    } COMPUTE # OF NUMERIC FIELD POSITIONS THAT CANNOT BE USED BY  5531  XXX XXXXX . XX
227E  D6 05                 SUI   H'05'  } CONVERTED #                                              V    5532
2280  F4 25 24              CP    X2425 IF ANY SPARE FIELD POSITIONS, FILL THEM WITH LEADING 'Ø' CHAR'S  %$  5533   # NEEDED FOR
2283  CD FA 24              CALL  X24FA CONVERT VALUE TO DIGIT STRING USING 2 BYTE SUBTRACT ROUTINE  M $  5534   INTEGER VALUE
2286  7B FROM 232A   X2286  MOV   A,E  PUT # OF NUMERIC FIELD POSITIONS TO RIGHT OF DP (INCLUDES DP POSITION) IN ACC.  5535
2287  B7                    ORA   A     TEST FIELD COUNT                                            7    5536
2288  CC C7 1A              CZ    X1AC7 IF COUNT IS Ø, REMOVE '.' CHAR IN BUFR BY DCX H (PUT IN BUFR BY  LG CALL 5537 2451 AT 2528, 2326 or 2432)
228B  3D                    DCR   A DECREMENT RIGHT-FIELD COUNT (COUNTS 1 POSITION FOR DP)          =    5538
228C  F4 25 24              CP    X2425 FILL BUFR WITH TRAILING 'Ø' CHAR'S - # OF 'Ø''S IN ACC FROM RIGHT #$  5539 FIELD COUNT
228F  E5             X228F  PUSH  H  SAVE PTR TO NXT BYTE AVAILABLE IN CHAR BUFR ON STACK - END OF STRING PTR  5540
2290  CD 8A 21              CALL  X218A PERFORM LEADING 'Ø' SUPPRESSION, INSERT '*', '$' + SIGN CHAR IN FRONT OF M!#-  5541 STRING (IF FORMATTED THAT WAY)
2293  E1                    POP   H  FETCH PTR TO BUFR INTO HL - POINTS TO NXT BYTE AFTER #-STRING           5542
2294  CA 99 22              JZ    X2299 JMP IF SIGN CHAR BELONGS IN FRONT OF #  (see 21FF-220Ø) -   J ''  5543 SIGN CHAR PUT IN FRONT OF # AT 21FF-220Ø
2297  70                    MOV   M,B STORE SIGN CHAR AFTER CONVERTED #                                       5544
2298  23                    INX   H  ADVANCE PTR TO NXT BYTE AVAILABLE IN CHAR BUFR                 Ø    5545
2299  36 00          X2299  MVI   M,H'00' STORE A NULL IN BUFR TO TERMINATE THE CHAR STRING         6    5546  ADJUST HL TO POINT TO 1ST CHAR OF CONVERTED
229B  21 20 04              LXI   H,X0420 LOAD HL WITH ADDR OF BYTE BEFORE 1ST CHAR OF CONVERTED #  !    5547  # STRING ACCORDING TO FORMAT REQUIREMENTS
229E  23             X229E  INX   H    ADVANCE PTR TO NXT CHAR IN BUFR                              Ø    5548       D = # OF DIGITS LEFT OF DP
229F  3A 05 03       X229F  LDA   X0305 LOAD ACC WITH LO-ORDER 8 BITS OF ADDR WHERE '.' CHAR IS STORED  5549
22A2  95                    SUB   L  SUBTRACT LO-ORDER BYTE OF ADDRESS OF 1ST CHAR IN BUFR                    5550
22A3  92                    SUB   D  SUBTRACT # OF NUMERIC FIELD POSITIONS TO LEFT OF DP (see 227C)           5551   ␣␣XXX . XX
22A4  C8        DONE <-     RZ    CONVERSION IS DONE IF # OF DIGITS TO LEFT OF DP MATCHES FORMAT  -> H>   5552     ↑       ↑
22A5  7E                    MOV   A,M  FETCH CHAR FROM CONVERTED # STRING INTO ACC                            5553   (HL)  (Ø306 Ø305)
22A6  FE 20                 CPI   A' '   } TEST CHAR         } TEST FOR ENOUGH CHAR'S TO FILL                 5554
22A8  CA 9E 22  <-          JZ    X229E  } JMP IF CHAR IS '␣' } LEFT SIDE OF DP FORMAT          J ''  5555
22AB  FE 2A                 CPI   A'*'   } TEST CHAR                                            *    5556
22AC  CA 9E 22  <-          JZ    X229E  } JMP IF CHAR IS '*'                                   J '' 5557 - THIS IS WHERE '%' IS PUT FOR
22B0  2B                    DCX   H  BACK-UP PTR TO LAST '␣' or '*' (CHAR BEFORE 1ST DIGIT, '$', or '+', '-')  5558   OVERFLOW INDICATION
22B1  E5                    PUSH  H  SAVE PTR ON STACK                                                    5559
22B2  F5             X22B2  PUSH  PSW SAVE CHAR ON STACK - ZERO FLAG SET (1ST TIME THRU ZERO FLAG RESET)  5560
22B3  01 B2 22              LXI   B,X22B2 } PUT ADDR 22B2 ON STACK FOR IMPLIED JMP              2 ''  5561
22B6  C5                    PUSH  B       }                                                     E    5562
22B7  D7                    RST   2 SCAN TO NXT NON-SPACE CHAR IN STRING - SET FLAGS - CHAR IN ACC  H   5563  PUT SIGN +/or $ ON STACK PRIOR TO
22B8  FE 2D                 CPI   A'-' } TEST CHAR + JMP IF CHAR IS '-'                         -    5564  REMOVING A LEADING Ø
22BA  C8        <-          RZ         }                                                        H    5565
22BB  FE 2B                 CPI   A'+' } TEST CHAR + JMP IF CHAR IS '+'                         +    5566
22BD  C8        <-          RZ         }                                                        H    5567
22BE  FE 24                 CPI   A'$' } TEST CHAR + JMP IF CHAR IS '$'                         $    5568
22C0  C8        <-          RZ         }                                                        H    5569
22C1  C1                    POP   B  REMOVE THE IMPLIED JMP ADDR FROM THE STACK                 A    5570
22C2  FE 30                 CPI   A'0' TEST CHAR                                                0    5571
22C4  C2 C8 22  <-          JNZ   X22C8 JMP IF CHAR IS NOT A LEADING 'Ø' - REMOVE CHAR'S FROM STACK + PUT X'' A 5572 '%' IN BUFR
22C7  23                    INX   H ADVANCE PTR TO BUFR }  ie. ␣␣+$ØØ1.234                      Ø    5573
22C8  D7                    RST   2 SCAN TO NXT CHAR        PUT ON STACK (HL) ↑ INX H ↑ CHAR FOUND BY RST 2  H  5574
22C9  D2 D8 22  <-          JNC   X22D8 JMP IF CHAR IS NOT A DIGIT <- AT 22D2                   RX '' 5575
22CC  2B                    DCX   H BACK-UP PTR (CANCELS INX H AT 22C7)  <- AT LEAST 1 DIGIT IN FRONT OF DP  5576
22CD  01 2B 77   DCX H MOV M,A  LXI  B,X772B } PUT SIGN CHAR +/or $ IN BUFR FROM STACK          +    5577
22D0  F1                    POP   PSW                                                                 5578
22D1  CA CE 22              JZ    X22CE } (PUT ON TOP OF 'Ø' FOUND AT 22C2 - ELIMINATES ONE LEADING Ø)  JN ''  5579
```

```
2204  C1              POP   B REMOVE PTR ADDR FROM STACK (NOT NEEDED HE.)          A
2205  C3 9F 22        JMP   X229F LOOP UNTIL ENOUGH LEADING Ø'S HAVE BEEN REMOVED   C
2208  F1       X2208  POP   PSW  REMOVE A CHAR FROM STACK                         JX    FORMAT FIELD OVERFLOW
2209  CA C8 22        JZ    X22D8 LOOP UNTIL CHAR REMOVED WITH ZERO FLAG RESET (SEE 22A0-22B2)
220C  E1              POP   H FETCH PTR TO CHAR POSITION IN BUFR BEFORE 1ST SIGNIFICANT CHAR (DIGIT ...)   6%
220D  36 25           MVI   M,A'%' PUT '%' CHAR IN BUFR TO INDICATE OVERFLOW       I
220F  C9              RET   CONVERSION DONE
----------------------------------------------------------------------------------------------
22E0  E5       X22E0  PUSH  H SAVE PTR TO LOC Ø423 ON STACK (LOC AFTER 1ST LEADING 'Ø')  FROM 226F   FLOATING-PT VALUE CONVERT
22E1  1F              RAR   TEST Ø1 BIT OF BIT-FLAG BYTE BY SHIFTING RIGHT 1 PLACE - CARRY BIT NOT SET UPON ENTRY   FORMATTED OUTPUT
22E2  DA 82 23  EXPONENTIAL FORMAT  JC  X2382 JMP IF DOING EXPONENTIAL FORMAT (Ø1 BIT SET IN BIT-FLAG BYTE - SEE Z p.#88)
22E5  CA FC 22        JZ    X22FC JMP IF VALUE TO CONVERT IS SNG-PREC VALUE (RAR NOT AFFECT Ø-FLAG-SEE 226C)  I%
22E8  11 49 25        LXI   D,X2549 LOAD DE WITH ADDR OF DBL-PREC VALUE = 1D16      M\
22EB  CD CC 18        CALL  X18CC 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 < 10^16 - # HAS LESS THAN 17 DIGITS TO PRINT   #.
                                                     MAGNITUDE OF # TO CONVERT TOO LARGE FOR FIELD (NON-EXPONENTIAL FORMAT)
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 2268)   M!
22F5  CD 81 21        CALL  X2181                                                  +
22F8  2B              DCX   H BACK-UP PTR TO CHAR BUFR (CHAR POSITION BEFORE SIGN CHAR OR '$')   6%
22F9  36 25           MVI   M,A'%' PUT '%' CHAR IN BUFR TO INDICATE OVERFLOW       I
22FB  C9              RET                                                          6
----------------------------------------------------------------------------------------------
22FC  01 0E B6 X22FC  LXI   B,XB60E } LOAD BCDE WITH SNG-PREC FLOATING-PT VALUE = 10^16   J
22FF  11 CA 13        LXI   D,X1BCA }                                              M!
2302  CD A1 18        CALL  X18A1 PERFORM SNG-PREC MAGNITUDE COMPARISON
2305  F2 F3 22        JP    X22F3 JMP IF #-TO-CONVERT ≥ 10^16 - 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             ACC HAS EXPONENT ADJUSTMENT  X10'S -> -ACC
230B  C4 D8 23        CNZ   X23D8 IF # ≠ Ø, ADJUST # IN HOLDING AREA SUCH THAT (SNG-PREC 10^5 ≤ # < 10^6) or (DBL-PREC 10^15 ≤ # < 10^16)  DXD   ÷10'S -> +ACC
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 2268)   A
2310  FA 20 23        JM    X2320 JMP IF # TO CONVERT WAS MULTIPLIED BY 10 (N TIMES) BY 23D8   -Ø   #'S ≥ 10^5 or 10^15 - NO FRACTIONS
2313  C5              PUSH  B PUT # OF NUMERIC POSITIONS TO LEFT OF DP (B) + # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) (C) ON STACK   E
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   }                                                   (+#)   %S
2318  F4 25 24        CP    X2425 IF ACC IS ≥ Ø, FILL BUFR WITH N LEADING Ø'S       M<S
2319  CC 3C 24        CALL  X243C COMPUTE VALUES FOR DECIMAL-PT + COMMA POSITION CTR'S   M_$
231F  CD 65 24        CALL  X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT   3   (Ø IN ACC (SEE 252E))
2321  B3              ORA   E PUT EXPONENT ADJUSTMENT IN ACC (SEE 2314) - SET FLAGS   053
2322  C4 35 24        CNZ   X2435 PUT TRAILING 'Ø''S IN CHAR BUFR (AS MANY AS NECESSARY)   3
2325  B3              ORA   E PUT EXPONENT ADJUSTMENT IN ACC - SET FLAGS            DD$
2326  C4 51 24        CNZ   X2451 IF CHAR ('Ø') PUT IN BUFR AT 2322, TEST IF COMMA OR DP GOES IN BUFR NEXT   Q
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   C
232A  C3 96 22        JMP   X2296 CONVERSION ALMOST DONE - CLEAN UP STRING, LEADING 'Ø' SUPPRESSION, INSERT SIGN CHAR
----------------------------------------------------------------------------------------------
232D  5F       X2320  MOV   E,A PUT EXPONENT ADJUSTMENT IN E (FROM CALL 23D8)       -   #'S < 10^5 or 10^15
232E  79              MOV   A,C PUT # OF NUMERIC POSITIONS TO RIGHT OF DP (INCLUDING DP) IN ACC   7
232F  B7              ORA   A TEST RIGHT-FIELD CTR                                  CU
2330  C4 05 20        CNZ   X2005 IF COUNT IS NOT Ø, DCR ACC BY 1 (ACCOUNTS FOR DP - TEMPORARY COMPUTATION SEE 2373)
2333  83        (-#)  ADD   E ADD EXPONENT ADJUSTMENT (# OF DIGITS OF ORIGINAL # THAT SHOULD BE ON RIGHT OF DP)
2334  FA 38 23        JM    X2338 IF SUM IS MINUS, # OF DIGITS IN ORIGINAL # TO RIGHT OF DP EXCEEDS FORMAT POSITIONS AVAILABLE   AFTER ADJUSTMENT
2337  AF              XRA   A CLEAR ACC - NO LOSS OF DIGITS OR ROUND-OFF INVOLVED - ENOUGH ROOM TO RIGHT OF DP FOR ALL DIGITS   E
2338  C5       X2338  PUSH  B SAVE BC ON STACK
2339  F5              PUSH  PSW SAVE LOST-DIGIT COUNT ON STACK (NEGATIVE #)         H
233A  FC 07 20 X233A  CM    X2007 DIVIDE # BY 10, ADD 1 TO ACC (PUSH 1 DIGIT TO RIGHT OF DP TO BE LOST IN CALL 2461)
233D  FA 3A 23        JM    X233A LOOP UNTIL ENOUGH LOST DIGITS HAVE BEEN PUSHED TO RIGHT OF DP   A
2340  C1              POP   B PUT LOST-DIGIT COUNT IN B
2341  79              MOV   A,E PUT EXPONENT ADJUSTMENT IN ACC (NEGATIVE #)
2342  90              SUB   B SUBTRACT LOST-DIGIT COUNT (NEGATIVE) or Ø             A
2343  C1              POP   B RESTORE BC FROM STACK
```

Margin notes:

- FIELD COUNT IN B / COUNT IN C: XXXXXXXXX.XXX — ORIGINAL # DDDDDD.DD — AFTER 23D8 DDDDDD.DDD ADJUSTMENT = 1(E) — # IN D (= Ø6 FOR SNG-PREC) ROUND-OFF OF 10 DIGITS
- RESULT: (B)-(D)-(E) = 3 — ØØØDDDDDDØ ORIGINAL # — IN SNG-PREC
- FIELD COUNT IN B / COUNT IN C: XXXXXXXXX.XXX — IN SNG-PREC — ORIGINAL # DDDD.DDDD — DDDDDD.DD ADJUSTMENT = +2 (E) — # IN D (Ø6 FOR SNG-PREC) ROUND OFF OF 10 DIGITS
- ORIGINAL # DD.DDDD ADJUST DDDDDD. — FORMAT XXXX.XXX — DIVIDE # ONCE DDDDD.D — THIS DIGIT LOST IN CONVERSION ROUNDED-OFF

```
2344  5F           MOV  E,A PUT NEW VALUE FOR EXPONENT ADJUSTMENT IN E (LOSING DIGITS IS A READJUSTMENT 5640 TO EXPONENT)
2345  92           ADD  D =Ø6 FOR SNG-PREC =1ØH FOR DBL-PREC                                          5641
2346  78           MOV  A,B PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN ACC                           5642
2347  FA 56 23     JM   X2356 JMP IF ENTIRE # IS SMALL ENOUGH TO FIT ENTIRELY TO RIGHT OF DP — ALL 6 V# 5643 DIGITS OF #
234A  92           SUB  D } COMPUTE # OF LEADING ZERO'S NEEDED TO FILL ALL NUMERIC POSITIONS TO LEFT OF 5644
234B  93           SUB  E }   DP     # IN ACC                                                         5645
234C  F4 25 24     CP   X2425 IF ACC ≥ Ø, FILL BUFR WITH N LEADING Ø'S                          %$    5646
234F  C5           PUSH B PUT # OF NUMERIC POSITIONS TO LEFT OF DP (B) & # OF POSITIONS TO RIGHT OF DP (INCLUDING E DP) 5647 ON STACK
2350  CD 3C 24     CALL X243C COMPUTE VALUES FOR DECIMAL-PT & COMMA POSITION CTR'S              M<$   5648
2353  C3 67 23     JMP  X2367 SKIP AROUND NXT INSTRUCTIONS                                     C #   5649
-------------------------------------------------------------------------------------------------------
2356  CD 25 24  X2356 CALL X2425 FILL ALL PLACES ON LEFT OF DP WITH N LEADING Ø'S              M%$   5650
2359  79           MOV  A,C PUT RIGHT NUMERIC POSITION COUNT IN ACC (SINCE C GETS DESTROYED BY NXT INSTR) 5651
235A  CD 55 24     CALL X2455 PUT '.' IN CHAR BUFR                                             MU$   5652 IF SNG-PREC
235D  4F           MOV  C,A PUT RIGHT NUMERIC POSITION COUNT IN C AGAIN                        O     5653 ØØØ.ØØXXXXXX
235E  AF           XRA  A } COMPUTE # OF Ø'S NEEDED TO RIGHT OF DP BEFORE # STRING             /     5654
235F  92           SUB  D }                                                                          5655
2360  93           SUB  E }                                                                          5656
2361  CD 25 24     CALL X2425 FILL BUFR WITH N MORE Ø'S                                        M%$   5657
2364  C5           PUSH B PUT # OF LEFT NUMERIC POSITIONS (B) & # OF RIGHT NUMERIC POSITIONS (C) E ON 5658 STACK
2365  47           MOV  B,A } CLEAR B & C - NO COMMA'S OR DP TO BE PUT IN BUFR IN THIS CASE    G     5659
2366  4F           MOV  C,A } (ACC = Ø FROM 2361)                                              O     5660
2367  CD 65 24  X2367 CALL X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT - ØØ M/N$ ACC 5661 (see 252E)
236A  C1           POP  B PUT # OF LEFT NUMERIC POSITIONS IN B, PUT # OF RIGHT NUMERIC POSITIONS (INCLUDING DP) 5662 IN C
236B  B1           ORA  C PUT RIGHT NUMERIC COUNT IN ACC - SET FLAGS (POSITIVE #)               1     5663
236C  C2 72 23     JNZ  X2372 JMP IF RIGHT NUMERIC COUNT ≠Ø                                    B #   5664
236F  2A 05 03     LHLD X0305 IF RIGHT NUMERIC COUNT = Ø, LAST CHAR IN BUFR IS '.', THIS IS WHERE *U PUT 5665 AT END OF BUFR (see 2299)
2372  83        X2372 ADD  E ADD EXPONENT ADJUSTMENT (NEGATIVE #)                                  5666
2373  3D           DCR  A DECREMENT RIGHT POSITION COUNT (ACCOUNTS FOR DP)                     =     5667
2374  F4 25 24     CP   X2425 IF ACC ≥ Ø, FILL BUFR WITH N TRAILING Ø'S                        %$    5668
2377  50           MOV  D,B PUT # OF LEFT NUMERIC POSITIONS IN D                               P     5669
2378  C3 9F 22     JMP  X229F CONVERSION ALMOST DONE - CLEAN UP STRING, LEADING ZERO SUPPRESSION, INSERT C SIGN 5670 CHAR
-------------------------------------------------------------------------------------------------------
2379  E5        X2379 PUSH H SAVE PTR TO LOC Ø423 ON STACK (LOC AFTER 1ST LEADING 'Ø')          U     5671 INTEGER # IN EXPONENTIAL FORMAT
237C  D5           PUSH D SAVE BIT-FLAG BYTE ON STACK (PUT IN D AT 21A4)                        M     5672
237D  CD 60 1C     CALL X1C60 CONVERT INTEGER VALUE TO SNG-PREC VALUE                           Q     5673
2380  D1           POP  D PUT BIT-FLAG BYTE IN D                                               /     5674
2381  AF           XRA  A CLEAR ACC TO FORCE JMP AT NXT INSTRUCTION                                  5675
2382  CA 88 23  X2382 JZ   X2388 JMP IF CONVERTING INTEGER OR SNG-PREC # TO EXPONENTIAL FORMAT J #   5676 FLOATING-PT # IN EXPONENTIAL FORMAT
2385  1E 10        MVI  E,H'10' PUT A DIGIT COUNT OF 1ØH (=16) IN E FOR DBL-PREC CONVERT             5677
2387  01 1E 06     LXI  B,X061E DUMMY PUT A DIGIT COUNT OF Ø6 IN E FOR SNG-PREC CONVERT              5678
238A  EF           RST  5 TEST SIGN OF #-TO-CONVERT IN HOLDING AREA                                  5679
238B  37           STC  SET CARRY - USED TO INDICATE #-TO-CONVERT IS Ø - CARRY IS CLEARED BY CALL 2370B IF 5680 IT IS EXECUTED
238C  C4 0A 23     CNZ  X230B IF # ≠ Ø, ADJUST # IN HOLDING AREA SUCH THAT (SNG-PREC 1Ø^5 ≤ # < 1Ø^6) or DBL-PREC DX# 5681 1Ø^15 ≤ # < 1Ø^16) ACC HAS EXPONENT ADJUSTMENT
238F  F1           POP  H PUT PTR TO LOC Ø423 IN HL (see 22EØ or 2378)                              5682
2390  C1           POP  B PUT # OF NUMERIC POSITIONS TO LEFT OF DP IN B, PUT # OF NUMERIC POSITIONS TO RIGHT OF DP 5683 (INCLUDING DP) IN C (see 2268)
2391  F5           PUSH PSW SAVE EXPONENT ADJUSTMENT & FLAG STATUS ON STACK                          5684
2392  79           MOV  A,C } PUT RIGHT NUMERIC COUNT IN ACC                                         5685
2393  B7           ORA  A   } & TEST IT                                                        7     5686
2394  F5           PUSH PSW SAVE FLAG STATUS FOR LATER USE                                           5687
2395  C4 05 20     CNZ  X2005 IF COUNT IS NOT Ø, DCR ACC BY 1 (ACCOUNTS FOR DP IN RIGHT NUMERIC COUNT) DU 5688
2398  80           ADD  B ADD LEFT NUMERIC COUNT TO ACC                                              5689
2399  4F           MOV  C,A STORE TOTAL NUMERIC POSITION COUNT IN C                            O     5691
239A  78           MOV  A,B } TEST BIT FLAG BYTE FOR TRAILING SIGN CHAR FORMAT                       5691
239B  E6 04        ANI  H'04' }   (Ø4 BIT SET IF TRAILING SIGN CHAR TO BE USED)                      5692
239D  FE 01        CPI  H'01' }                                                                      5693
239F  9F           SBB  A   } RESULT: ACC = -1 IF LEADING SIGN  ACC = Ø IF TRAILING SIGN (TRAILING 5695 SIGN NOT PART OF NUMERIC POSITION
23A0  57           MOV  D,A SAVE THIS SIGN COUNT IN D. (-1 FOR LEADING SIGN Ø OTHERWISE)             5696 COUNTS (see p 88
23A1  81           ADD  C } COMPUTE # OF NUMERIC POSITIONS (TOTAL)   XXX.XX                    O     5697
23A2  4F           MOV  C,A } STORE THIS NEW VALUE BACK INTO C                                       5698
23A3  93           SUB  E SUBTRACT DIGIT COUNT FROM TOTAL COUNT (DIGIT COUNT = Ø6 FOR SNG-PREC, =1ØH 5699 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 ...))
```

ACC ≠ Ø ZERO FLAG SET N

SNG-PREC (see 2366)

```
23A5  C5              PUSH B  SAVE BC ON STACK (THIS PAIR NOT SAVED ON STACK BY 23D7)
23A6  FC D7 20  X23A6 CM   X20D7 IF ACC HAS -#, DIVIDE ADJUSTED # (see 23BC) BY 10 - ELIMINATES 1 DIGIT
23A9  FA A6 23        JM   X23A6 LOOP UNTIL ACC=0, N DIGITS HAVE BEEN REMOVED FROM # BEFORE
23AC  C1              POP  B RESTORE BC FROM STACK                                  CONVERSION BY MAKING THOSE DIGITS TO PART OF A FRACTION.
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      (+# >0 IF EXTRA 0'S NEEDED)
23AF  F5              PUSH PSW SAVE FLAG STATUS & LOST DIGIT COUNT (GAINED DIGIT COUNT IF >0)
2390  FA B4 23        JM   X23B4 JMP IF NOT ALL DIGITS IN # TO BE PRINTED AFTER CONVERSION
2393  AF              XRA  A  CLEAR ACC - NO DIGITS LOST
2394  2F      X23B4   CMA  } FORM 2'S COMPLEMENT OF # OF DIGITS LOST (+# IN ACC NOW)
2395  3C              INR  A} SAME AS # OF TIMES 23A6-23A9 LOOP EXECUTED
2396  80              ADD  B  ADD LEFT NUMERIC COUNT TO ACC
2397  3C              INR  A  ADD 1 TO ACC (COMPENSATES FOR DCR B AT START OF LOOP AT 2451)
2398  82              ADD  D  ADD SIGN CHAR COUNT (-1 FOR LEADING SIGN, 0 OTHERWISE)
2399  47              MOV  B,A PUT VALUE FOR DECIMAL-PT POSITION CTR IN B (USED AT 2451)
239A  0E 00           MVI  C,H'00 SET COMMA POSITION CTR TO 0 (NO COMMA'S ALLOWED IN EXPONENTIAL
239C  CC 65 24        CALL X2465 CONVERT ADJUSTED MAGNITUDE OF FLOATING-PT # TO CHAR STRING EQUIVALENT
239F  F1              POP  PSW FETCH FLAG STATUS & LOST (GAINED) DIGIT COUNT FROM STACK
23C0  F4 2E 24        CP   X242E IF # IN ACC >0, ADD N TRAILING 0'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 0, REMOVE '.' CHAR IN BUFR BY DCX H (PUT IN LGBUFR
23C8  F1              POP  PSW FETCH EXPONENT ADJUSTMENT & FLAG STATUS FROM STACK
23C9  DA CF 23        JC   X23CF JMP IF # TO CONVERT IS 0 - see 23BB
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 9F 22        JMP  X229F 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 X10, POSITIVE COUNT FOR ÷10)
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 (10^16 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 = 10^5
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 = 10^15
23F1  CD 0C 19        CALL X190C PERFORM DBL-PREC MAGNITUDE COMPARISON
23F4  F2 07 24  X23F4 JP   X2407 JMP IF (SNG-PREC # ≥ 10^5) or (DBL-PREC # ≥ 10^16)
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 10 - DECR ACC BY 1
23FB  F5              PUSH PSW SAVE EXPONENT-ADJUST CTR ON STACK
23FC  C3 DE 23  X10 loop JMP X23DE LOOP UNTIL (SNG-PREC # ≥ 10^5) or (DBL-PREC VALUE ≥ 10^16)
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 10 - INCR ACC BY 1
2403  F5              PUSH PSW SAVE EXPONENT-ADJUST CTR ON STACK
2404  CD 0A 24  ÷10 loop CALL X240A TEST IF (SNG-PREC # < 10^6) or (DBL-PREC # < 10^16) - 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 = 10^6
2411  11 F8 23        LXI  D,X23F8 }
```

```
E   5700
    5701 ON THE LO END OF ORIGINAL #
    5702
    5703
A:  5704 CONVERTED)
E   5705
    5706 & SNG-PREC
    5707      FORMAT   XX.XXE±XX
/   5708
    5709 CONVERSION IS 6 OR 16 DIGITS
    5710 OF LEADING 0'S TO SKIP BEFORE INSERTING
    5711
<   5712
    5713
G   5714
    5715 FORMAT)
    5716
M $ 5717
.$  5718 NECESSARY
    5719 (SIGN CHAR ALREADY COUNTED - 239F)
    5720
    5721 BY CALL 2451 AT 2528, 2326, or 2432)
    5722 IF ORIGINAL # IS .XXXXXX · 10^N
ZO# 5723 AFTER ADJUSTMENT # IS XXXXXX. · 10^(N-6)
    5724 EXPONENT ADJUSTMENT IS N-6
    5725 DESIRED FORMAT TO PRINT
    5726      XX.XXXX
E   5727      K DIGITS
MC" 5728                M=6 SNG-PREC
    5729      M DIGITS  M=16 DBL-PREC
Q   5730
I   5731
D   5732 ADJUST FLOATING-PT # IN HOLDING AREA
    5733 SUCH THAT
    5734      SNG-PREC # 10^5 ≤ # < 10^6
MI$ IT 5735 IS
    5736      DBL-PREC # 10^15 ≤ # < 10^16
#   5737
C   5738 SNG-PREC TEST # ≥ 10^5
O   5739
M!  5740
C # 5741
1%  5742 DBL-PREC TEST # ≥ 10^15
M\  5743
$   5744
    5745
MJ  5746
    5747
C^# 5748
    5749
MW  5750
    5751
M/$ loop 5752 to 23FF to ÷10 til it is
    5753
Q   5754
I   5755
    5756
$   5757 TEST IF
    5758      SNG-PREC # < 10^6
#   5759      DBL-PREC # < 10^16
          NORMAL RETURN IF IT IS
          ÷10 IF IT ISN'T - LOOP TIL IT ...
```

```
2414  CD A1 19           CALL X1DA1 PERFORM SNG-PREC MAGNITUDE COMPARISON - TEST IF VALUE IN HOLDING AREA IS > 10^6
                                                                         RZ  # = CONSTANT
                                                                         RNC # > CONSTANT
                                                                         RC  # < CONSTANT
2417  C3 20 24           JMP  X2420 SKIP 2 INSTRUCTIONS
241A  11 39 25    X241A  LXI  D,X2539 LOAD DE WITH ADDR OF DBL-PREC CONSTANT = 10^16
241D  CD DC 1B           CALL X1BDC MOVE CONSTANT TO DBL-PREC TEMP STORAGE AREA - PERFORM DBL-PREC MAGNITUDE COMPARISON
2420  E1          X2420  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 # OF ÷10 PERFORMED
2424  E9                 PCHL      RETURN IF # IS LESS THAN
-----------------------------------------------------------------------------
2425  B7          X2425  ORA  A TEST # IN ACC                         STORE N LEADING OR TRAILING Ø'S
2426  C8          X2426  RZ   RETURN IF DONE, POSITION CTR IS ZERO    IN BUFR, N IS # IN ACC
2427  3D                 DCR  A DECREMENT POSITION CTR
2428  36 30              MVI  M,A'0' STORE A 'Ø' IN BUFR
242A  23                 INX  H ADVANCE PTR TO NXT LOC IN BUFR
242B  C3 26 24           JMP  X2426 LOOP UNTIL CTR IS Ø
-----------------------------------------------------------------------------
242E  C2 35 24    X242E  JNZ  X2435                                   STORE N LEADING OR TRAILING Ø'S
2431  C8          X2431  RZ   RETURN IF DONE, POSITION CTR IS ZERO    IN BUFR, N IS # IN ACC
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 'Ø' 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 Ø
-----------------------------------------------------------------------------
243C  7B          X243C  MOV  A,E PUT EXPONENT-ADJUSTMENT IN ACC (see 2314) (E is + #)
                                                                  COMPUTE POSITION-CTR VALUES FOR
243D  82                 ADD  D ADD DIGIT COUNT (see 230F Ø6 FOR SNG-PREC, see 22EE 10 FOR DBL-PREC)
                                                                  DECIMAL-PT POSITION + COMMA POSITION
243E  3C                 INR  A ADD 1 TO ACC (COMPENSATES FOR DCR B AT START OF LOOP AT 2451)
                                                                  (IF ',' FORMAT USED) IN CONVERTED-# STRING
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)
244A  4F                 MOV  C,A   }
2449  3A CD 03    X2449  LDA  X03CD LOAD ACC WITH BIT-FLAG BYTE (see 22Ø2)
                                                                  ZERO COMMA POSITION CTR IF NOT USING
                                                                  ',' FORMAT - INHIBITS COMMA'S AT 245D
244C  E6 40              ANI  A'@' TEST 4Ø BIT (SET IF COMMA'S TO BE PUT TO LEFT OF DECIMAL-PT - see p. 85)
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               PRINT USING FORMATTER
2452  C2 5D 24           JNZ  X245D. IF NOT INSERTING A DECIMAL-PT, TRY FOR A COMMA
                                                                  INSERT DECIMAL POINT
2455  36 2E       X2455  MVI  M,A'.' PUT DECIMAL-PT IN CHAR BUFR      WHEN B HAS COUNTED TO Ø
2457  22 05 03           SHLD X0305  SAVE ADDR OF DECIMAL-PT IN FLAG AREA
                                                                  CANCEL ALL COMMA'S AFTER '.'
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          X245D  DCR  C      DECREMENT COMMA CTR                    INSERT COMMA WHEN C
245E  C0                 RNZ         RETURN IF NOT READY TO INSERT A COMMA  HAS COUNTED TO Ø
245F  36 2C              MVI  M,A',' PUT COMMA IN CHAR BUFR                 INSERT COMMA EVERY 3RD
2461  23                 INX  H      ADVANCE CHAR BUFR PTR                  DIGIT THEREAFTER
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          CONVERT ADJUSTED MAGNITUDE
2466  F7                 RST  6 TEST VALUE TYPE OF #-TO-BE-CONVERTED  OF FLOATING-PT # TO CHAR
2467  E2 32 24           JPO  X2432 JMP IF VALUE IS SNG-PREC TYPE     STRING EQUIVALENT
246A  C5                 PUSH D SAVE DEC-PT + COMMA CTR'S ON STACK
246B  E5                 PUSH H SAVE PTR TO CHAR BUFR ON STACK
246C  CD 91 19           CALL X1C91 PUT DBL-PREC VALUE TO BE CONVERTED IN TEMP STORAGE AREA Ø41B - Ø41F
                                                                  - # HAS BEEN ADJUSTED SUCH THAT
                                                                  10^15 ≤ # < 10^16.
246F  21 41 25           LXI  H,X2541 LOAD HL WITH ADDR OF DBL-PREC CONSTANT = .5
2472  CD 8B 1B           CALL X1B8B MOVE DBL-PREC CONSTANT TO HOLDING AREA Ø4ØE - Ø415
2475  CD 4A 1E           CALL X1E4A ADD .5 TO # TO BE CONVERTED USING DBL-PREC ADD ROUTINE - ROUNDOFF OF LO-ORDER DIGIT
2478  AF                 XRA  A CLEAR CARRY BIT - SIGNIFIES SPECIAL RETURN (see 1D23)
2479  CC 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
```

```
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 'Ø'
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
2497  E1                POP  H  RESET HL TO POINT TO START OF DECIMAL CONSTANT
2498  3E AE             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  F5                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 ØØØE TO HOLDING AREA Ø412
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 19          CALL X188D ADD .5 TO SNG-PREC # IN HOLDING AREA
24B7  3C                INR  A ADJUST EXPONENT (IN ACC FROM 188D) - SET ACC TO NON-ZERO VALUE - DOES NOT
24B8  CD AF 1C          CALL X1CAF PERFORM SNG-PREC INT FCN ON # IN HOLDING AREA  ROUNDOFF OF
24BB  CD 46 18          CALL X1846 STORE #-IN-BCDE IN HOLDING AREA Ø412-Ø415
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
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 CONSTAINT TABLE IN HL
24D0  06 2F             MVI  B,A'/' PRESET REG B TO 1 LESS THAN 'Ø'
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
#   5840
    5841
A   5342
=   5343
B $ 5844
E   5845
    5846
! AREA 5847
MC Ø412 5848
C $ 5849
E   5850
    5851
M   5852
    5853 APPEAR TO AFFECT VALUE
    5854
MF  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
MQ) 5866
    5867
/   5868
    5869
    5870
    5871
    5872
#   5873
    5874
    5875
W   5876
Ø   5877
    5878
    5879
```

121

```
                              } PART OF HI BYTE SUBTRACTION
2400  4F                 MOV  C,A                                                          O
240E  2B                 DCX  H  } BACK-UP HL PTR TO POINT TO LO BYTE OF 3 BYTE DECIMAL    +
240F  2B                 DCX  H  }                                               CONSTANT  +
24E0  D2 02 24           JNC  X2402 IF # IN C D E IS STILL POSITIVE, PERFORM ANOTHER SUBTRACTION  RR$
24E3  CC 43 19           CALL X1943 ADD CONSTANT TO # IN C,D E (3 BYTE ADD) MAKES # IN HC     C D E POSITIVE AGAIN
24E6  23                 INX  H  ADVANCE PTR TO POINT TO LO BYTE OF NEXT DECIMAL CONSTANT  #  CONSTANT STORED AT LOC IN HL
24E7  CC 46 10           CALL X1046 STORE 3 BYTE # IN C D E AT LOC Ø414 Ø413 Ø412   MF  } (B DON'T CARE)
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                     A
24F0  DA C4 24           JC   X24C4 IF CARRY IS SET (1ST TIME THRU) LOOP FOR 2ND PASS THRU  ZD$   LOOP
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                     C %
-----------------------------------------------------------------------------------------------
24FA  05        X24FA    PUSH D  SAVE CONTENTS OF DE ON STACK                     U     INTEGER CONVERSION
24FC  11 90 25           LXI  D,X2590 PUT START ADDR OF DECIMAL CONSTANT TABLE IN DE  %  BINARY TO DECIMAL (ASCII)
                                                        (2-BYTE CONSTANTS)
24FE  3E 05              MVI  A,H'05' PUT DIGIT-CONVERT LOOP COUNT IN ACC          >
2500  CD 51 24  →X2500   CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY  MQS
2503  C5                 PUSH B  SAVE DEC-PT + COMMA CTR'S ON STACK                E
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   N
2508  23                 INX  H  ADVANCE PTR TO CONSTANT TABLE      } CONSTANT IN BC #
2509  46                 MOV  B,M FETCH HI BYTE OF DECIMAL CONSTANT }              F
250A  C5                 PUSH B PUT DECIMAL CONSTANT ON STACK                      E
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 'Ø'               /
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 }  ---------  --      # IN HL IS SMALLER THAN CONSTANT
2517  7C                 MOV  A,H }   H         L
2518  9A                 SBB  D   }  THIS LOOP COUNTS THE # OF TIMES THE CONSTANT CAN
2519  67                 MOV  H,A }  BE SUBTRACTED FROM THE # TO CONVERT. REG B IS THE CTR   (ASCII #)
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            C
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              A
2527  3D                 DCR  A DECREMENT LOOP CTR                                  =
2528  C2 00 25     ←     JNZ  X2500 WHEN Ø, 5 ASCII DIGITS HAVE BEEN EXTRACTED FROM  B %   # TO BE CONVERTED
252B  CD 51 24           CALL X2451 PUT DECIMAL-PT OR COMMA IN CHAR BUFR IF NECESSARY  MQS
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                        Q
2530  C9                 RET  END OF CONVERSION                                      I
-----------------------------------------------------------------------------------------------
2531  FD        X2531    DATA H'FD' }
2532  FF                 RST  7     } = 10^15 DBL-PREC CONSTANT
2533  9F                 SBB  A     }
```

```
2534  31 A. 5F        LXI  SP,X5FA9   } = 10^15 DBL-PREC CONSTANT
2537  63              MOV  H,E        }                                  1)_   5940
2538  B2              ORA  D          }                                        5941
2539  FE FF     X2539 CPI  H'FF'      }                                  2     5942
253A  03              INX  B          }                                        5943
253C  BF              CMP  A          } = 10^16 DBL-PREC CONSTANT              5944
253D  C9              RET             }                                  ?     5945
253E  1B              DCX  D          }                                  I     5946
253F  0E B6           MVI  C,H'B6'    }                                        5947
                                                                         6     5948
2541  00        X2541 NOP             }                                        5949
2542  00              NOP             } FLOATING POINT                         5950
2543  00              NOP             } DOUBLE-PRECISION                       5951
2544  00              NOP             } VARIABLE = .5                          5952
2545  00        X2545 NOP             }                                        5953
2546  00              NOP             } FLOATING POINT                         5954
2547  00              NOP             } SINGLE-PRECISION                       5955
2548  80              ADD  B          } VARIABLE = .5                          5956
2549  00        X2549 NOP             FLOATING POINT DOUBLE PRECISION CONSTANT 5957
254A  00              NOP               = 1D+16                                5958
254B  04              INR  B                                                   5959
254C  BF              CMP  A                                                   5960
254D  C9              RET                                                ?     5961
254E  1B              DCX  D                                             I     5962
254F  0E B6           MVI  C,H'B6'                                       6     5963
2551  00        X2551 NOP             }                                        5964
2552  80              ADD  B          } CONSTANT = 1E15                        5965
2553  C6 A4           ADI  H'A4'      }                                  F$    5966
2555  7E              MOV  A,M        }                                        5967
2556  8D              ADC  L          }                                        5968
2557  03              INX  B          }                                        5969
2558  00              NOP             }                                        5970
2559  40              MOV  B,B        } CONSTANT = 1E14                  @     5971
255A  7A              MOV  A,D        }                                        5972
255B  10              DATA H'10'      }                                        5973
255C  F3              DI              }                                        5974
255D  5A              MOV  E,D        }                                  Z     5975
255E  00              NOP             }                                        5976
255F  00              NOP             }                                        5977
2560  A0              ANA  B          } CONSTANT = 1E13                        5978
2561  72              MOV  M,D        }                                        5979
2562  4E              MOV  C,M        }                                  N     5980
2563  18              DATA H'18'      }                                        5981
2564  09              DAD  B          }                                        5982
2565  00              NOP             }                                        5983
2566  00              NOP             }                                        5984
2567  10              DATA H'10'      } CONSTANT = 1E12                        5985
2568  A5              ANA  L          }                                  %     5986
2569  D4 E8 00        CNC  X00E8      }                                  T     5987
256C  00              NOP             }                                        5988
256D  00              NOP             }                                        5989
256E  E8              RPE             } CONSTANT = 1E11                        5990
256F  76              MOV  M,M        }                                        5991
2570  48              MOV  C,B        }                                  H     5992
2571  17              RAL             }                                        5993
2572  00              NOP             }                                        5994
2573  00              NOP             }                                        5995
2574  00              NOP             }                                        5996
2575  E4 0B 54        CPO  X540B      } CONSTANT = 1E10                  T     5997
2578  02              STAX B          }                                        5998
2579  00              NOP             }                                        5999
```

```
257A 00          NOP
257B 00          NOP
257C CA 9A 39    JZ   X399A     } CONSTANT = 1E9          J ;
257F 00          NOP
2580 00          NOP
2581 00          NOP
2582 00          NOP
2583 E1          POP  H         } CONSTANT = 1E8
2584 F5          PUSH PSW
2585 05          DCR  B
2586 00          NOP
2587 00          NOP
2588 00          NOP
2589 80          ADD  B         } CONSTANT = 1E7
258A 96          SUB  M
258B 98          SBB  B
258C 00          NOP
258D 00          NOP
258E 00          NOP
258F 00          NOP
2590 40          MOV  B,B       } CONSTANT = 1,000,000 = 1E6      B
2591 42          MOV  B,D
2592 0F.         RRC
2593 00          NOP
2594 00          NOP
2595 00          NOP
2596 00          NOP
2597 A0    X2597 ANA  B         3 BYTE CONSTANT = 1000000
2598 86          ADD  M
2599 01 10 27    LXI  B,X2710   3 BYTE CONSTANT = 10000
259C 00          NOP
259D 10    X259D DATA H'10'     } LO BYTE
259E 27          DAA            } HI BYTE     INTEGER CONSTANT = 10000
259F E8          RPE            INTEGER CONSTANT = 1000
25A0 03          INX  B
25A1 64          MOV  H,H       INTEGER CONSTANT = 100
25A2 00          NOP
25A3 0A          LDAX B         INTEGER CONSTANT = 10
25A4 00 INTEGER CONSTANT=1 NOP
25A5 01 00 21 LXI H,1B13  LXI  B,X2100  } LOAD HL WITH ADDR 1B13 - ROUTINE TO NEGATE FLOATING-PT
25A8 13                   INX  D        } # IN HOLDING AREA
25A9 1B                   DCX  D        }
25AA E3                   XTHL          } PUT RETURN ADDR IN HL, PUT ADDR 1B13 ON STACK
25AB E9                   PCHL          } EXECUTE RETURN TO CALLING ROUTINE (CALL 25A7) BY PUTTING RETURN ADDR IN PC
25AC CD 36 1B             CALL X1B36  PUT SNG-PREC # IN HOLDING AREA ONTO STACK (ARG TO SQR)   M6
25AF 21 45 25             LXI  H,X2545  LOAD HL WITH ADDR OF SNG-PREC CONSTANT = .5            !E%
25B2 CD 43 1B             CALL X1B43  PUT SNG-PREC CONSTANT = .5 IN HOLDING AREA               MC
25B5 C3 99 25             JMP  X2588  SKIP NXT INSTRUCTION                                     C;%
25B8 CD 45 1C       X2588 CALL X1C45  CALL CSGN ROUTINE - CONVERT VALUE IN HOLDING AREA TO SNG-PREC  ME
25B8                X2588 POP  B    } PUT ARGUMENT TO SQR IN BCDE FROM STACK                    A
25BC C1                   POP  D    }   also Y                                                  0
25BD EF                   RST  5  FLOATING-PT SIGN TEST OF # IN HOLDING AREA
25BE CA F7 25  X=0 Y^X=1  JZ   X25F7  IF EXPONENT (X IN Y^X) IS ZERO, EVALUATE Y^0 = 1         J %
25C1 78                   MOV  A,B  } TEST EXPONENT OF SNG-PREC # IN BCDE
25C2 B7                   ORA  A    }                                                          7
25C3 CA 03 19  <DONE      JZ   X1903  IF # IN BCDE IS 0, CHANGE # IN HOLDING AREA TO 0 -       JSQR
25C6 D5        Y=0        PUSH D    } PUT SNG-PREC # IN BCDE (ARG TO SQR) ON STACK             UDONE
25C7 C5        Y^X=1      PUSH B    }   Y                                                      E
25C8 79                   MOV  A,C  } TEST SIGN BIT OF ARG TO SQR
25C9 F6 7F                ORI  H'7F'}
```

6000 6001 6002 6003 6004 6005 6006 6007 6008 6009 6010 6011 6012 6013 6014 6015 6016 6017 6018 6019 6020 6021 6022 6023 6024 6025 6026 6027 6028 6029 6030 6031 6032 6033 6034 6035 6036 6037 6038 6039 6040 6041 6042 6043 6044 6045 6046 6047 6048 6049 6050 6051 6052 6053 6054 6055 6056 6057 6058 6059

SQR

$y^x = EXP(X * LNY)$

```
25C8  CD 51 1B        CALL X1351 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  CC 03 19        CALL X1903 COMPUTE PRODUCT X*LNY, THEN COMPUTE EXP TO FINISH
2527  CD 36 19  X25F7 CALL X1D36 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       } = 1.2^(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  D,H
2569  2E 94           MVI  L,H'94' } SINGLE PRECISION CONSTANT
256B  74              MOV  M,H     } = -.0000141316 ≃ 1/7076
256C  70              MOV  M,B
256D  4F              MOV  C,A     } SINGLE PRECISION CONSTANT
256E  2E 77           MVI  L,H'77' } = .0000132988 ≃ 1/752
2570  6E              MOV  L,M
2571  02              STAX B       } SINGLE PRECISION CONSTANT
                                   } = -.0000830136
```

```
MQ   6050
_%   6061
U    6062
E    6063
ME   6064
A    6065
O    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
MG   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
8    6112
.    6113
     6114
     6115
0    6116
.    6117
     6118
     6119
```

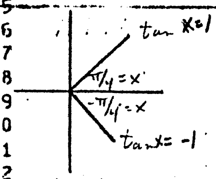
```
2642  88                    ADC  B      } PART OF 3RD SINGLE PRECISION
2643  7A                    MOV  A,D    } CONSTANT = -.ØØ83Ø136 ≃ 1/12Ø
2644  E6 A0                 ANI  H'A0'  } SINGLE PRECISION CONSTANT
2646  2A 7C 50              LHLD X507C  } = .Ø4165744 = 1/24
2649  AA                    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    } = .5
264F  7F                    MOV  A,A    }
2650  00                    NOP         }
2651  00                    NOP         } SINGLE PRECISION CONSTANT
2652  80                    ADD  B      } = 1
2653  81                    ADD  C      }
2654  00                    NOP         }
2655  00                    NOP         } SINGLE PRECISION CONSTANT
2656  00                    NOP         } = 1
2657  81                    ADD  C      }
2658  CD 36 18    X2658     CALL X1836  PUT ARG IN HOLDING AREA ONTO STACK
265B  11 09 10              LXI  D,X1009} PUT ADDR 1009 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 03 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  06 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 03 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 70 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  01 35 98    X268A     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 03 19              CALL X1903 PERFORM SNG-PREC MULTIPLY   }
269C  01 2A 68              LXI  B,X682B} LOAD BCDE WITH SNG-PREC  } ADD 3.92768E-Ø8 TO ABOVE RESULT
269F  11 46 81              LXI  D,X8146} 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  78                    MOV  A,E} SWAP MSB + LSB OF #
26A9  59                    MOV  E,C}
```

```
6120
6121
6122
*P  6123
*   6124
*   6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
M6  6139
9   6140
U   6141
    6142
MQ  6143
MS  6144
    6145
M6  6146  ↓ POLYNOMIAL EVALUATOR
    6147    TABLE OF COEFFICIENTS
#   6148    ITERATION CTR - 1 BYTE
MC  6149    AN
    6150    AN-1      } SNG-PREC
A   6151    :         } CONSTANTS
Q   6152    A1        }
=   6153    A0        }
H   6154
U   6155
E   6156
    6157
    6158  RESULT IN HOLDING AREA
MS  6159
    6160
MT  6161  ((AN*X)+AN-1)*X...+A0
    6162
M   6163
    6164
C & 6165  ANX^N+...A1X+A0
    6166  RND
 %& 6167
!:& 6168
MC  6169
H   6170
5   6171
 D  6172
MS  6173
(   6174
F1  6175
M   6176
MQ#) 6177  - HL = Ø416
    6178
Y   6179
```

B C D E ← LSB BYTE (ACC)

```
                                 } SWAP MSB + LSB OF #
26AA  4F                 MOV  C,A                                                        O
26AB  36 80              MVI  M,H'80' SET SIGN BIT IN LOC Ø416 TO 1 (+ SIGN) see 1873 FOR DERIVATION + USE OF Ø416
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 1851 AT 26A5        F
26AF  36 80              MVI  M,H'80' SET EXPONENT TO 2^-1                                6
26B1  CD E0 18           CALL X18ED NORMALIZE # IN BCDE, STORE RESULT IN HOLDING AREA     M
26B4  21 BA 26           LXI  H,X26BA LOAD HL WITH ADDR OF LSB OF WHERE RANDOM # IS TO BE SAVED !!&
26B7  C3 5D 19           JMP  X195D STORE RANDOM # BACK INTO RANDOM# HOLDING AREA FROM SNG-PREC   C}   HOLDING AREA
---------------------------------------------------------------------------------------------------
26BA  52        X26BA    MOV  D,D  } SNG-PREC CONSTANT = .811635                          R
26BB  C7                 RST  0    }                                                      G
26BC  4F                 MOV  C,A  } RANDOM # HOLDING AREA                                O
26BD  80                 ADD  B    }
---------------------------------------------------------------------------------------------------
26BE  21 04 27  X26BE    LXI  H,X2704 LOAD HL WITH ADDR OF CONSTANT = π/2                 !     COS = SIN(ARG + π/2)
26C1  CD 90 18           CALL X1890 FETCH CONSTANT INTO BCDE, ADD TO ARGUMENT, THEN COMPUTE SIN   M
---------------------------------------------------------------------------------------------------
26C4  CD 36 13  X26C4    CALL X1B36 PUT ARG (# IN HOLDING AREA) ONTO STACK                M6    SIN
26C7  01 49 83           LXI  B,X8349 } LOAD BCDE WITH SNG-PREC CONSTANT = 2π             I
26CA  11 DB 0F           LXI  D,X0FDB }                                                   {
26CD  CD 46 18           CALL X1946 PUT CONSTANT IN HOLDING AREA                          MF
26D0  C1                 POP  B } FETCH ARG FROM STACK INTO BCDE                          A
26D1  D1                 POP  D }                                                         Q
26D2  CD 2E 1A           CALL X1A2E PERFORM SNG-PREC DIVIDE, COMPUTE (ARG/2π)             M.
26D5  CD 36 1B           CALL X1B36 PUT RESULT OF DIVISION ONTO STACK                     M6
26D8  CD C5 1C           CALL X1CC5 PERFORM INT FUNCTION ON RESULT OF DIVISION            ME
26DB  C1                 POP  B } FETCH (ARG/2π) FROM STACK                               A.
26DC  D1                 POP  D }                                                         Q
26DD  CD 99 1A           CALL X1899 PERFORM SNG-PREC SUBTRACT, COMPUTE FRACTIONAL REMAINDER OF (ARG/2π)   M
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   M
26E6  EF                 RST  5 PERFORM FLOATING-PT SIGN TEST ON RESULT
26E7  37                 STC  SET CARRY TO SKIP CALL AT 26FB IF RESULT ≥ .25              7
26E8  F2 F0 26           JP   X26F0                                                       &
26EB  CD 80 18           CALL X1880 ADD .5 TO RESULT IN HOLDING AREA                      M
26EE  EF                 RST  5 PERFORM FLOATING-PT SIGN TEST ON NEW RESULT
26EF  B7                 ORA  A SET FLAGS - NC IF NEW RESULT ≥ Ø                          7
26F0  F5        X26F0    PUSH PSW SAVE FLAG STATUS ON STACK
26F1  F4 13 1B           CP   X1B13 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          M
26FA  F1                 POP  PSW FETCH FLAG STATUS FROM STACK
26FB  D4 13 1B           CNC  X1B13 IF CARRY NOT SET, NEGATE # IN HOLDING AREA            T
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  CX&
---------------------------------------------------------------------------------------------------
2704  DB 0F     X2704    IN   H'0F' } SINGLE PRECISION CONSTANT                           {
2706  49                 MOV  C,C   }                                                     I
2707  81                 ADD  C     } = 1.57Ø8 = π/2
---------------------------------------------------------------------------------------------------
2708  00        X2708    NOP  } SINGLE PRECISION CONSTANT
2709  00                 NOP  }
270A  00                 NOP  }
270B  7F                 MOV  A,A } = .25
---------------------------------------------------------------------------------------------------
270C  05        X270C    DCR  B CONSTANT TO INDICATE 5 SINGLE PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE
---------------------------------------------------------------------------------------------------
270D  DA                 CMP  D                                                           W
270E  07                 RST  2        = 39.71Ø7
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                                                           X
---------------------------------------------------------------------------------------------------
2715  58                 MOV  E,B                                                         4
2716  34                 INR  M        = 81.6Ø22                                          #
2717  23                 INX  H
2719  87                 ADD  A
```

```
2719  E0                RPO                                                              )
271A  50                MOV   E,L      = -41.3417                                        %
271B  A5                ANA   L
271C  86                ADD   M
2710  DA 0F 49          JC    X490F                                                      Z I
2720  33                ADD   E        = 6.28319 = 2π
2721  CD 36 19          CALL  X1836 PUT ARG (IN HOLDING AREA) ONTO STACK                 M6
2724  CD C4 26          CALL  X26C4 COMPUTE SIN(ARG) - RESULT IN HOLDING AREA            MD&     TAN
2727  C1                POP   B } FETCH ARG FROM STACK, HALF OF # PUT IN HL DUE TO FORMAT  A
2728  E1                POP   H } OF 1836 ROUTINE
2729  CD 36 18          CALL  X1836 PUT PARTIAL RESULT SIN(ARG) ON STACK                 M6
272C  EB                XCHG  PUT THE LO 2 BYTES OF ARG IN DE NOW - STANDARD FORMAT FOR SNG-PREC #  IN BCDE
2720  CD 46 18          CALL  X1846 PUT # IN BCDE BACK INTO HOLDING AREA (ARG IN HOLDING AREA   MF
2730  CD 8E 26          CALL  X268E COMPUTE COS(ARG) - RESULT IN HOLDING AREA            H>&
2733  C3 2C 1A          JMP   X1A2C COMPUTE SIN(ARG)/COS(ARG) - END OF TAN - RESULT IN HOLDING AREA  C.
2736  EF       X2736    RST   5  PERFORM SIGN TEST ON ARGUMENT TO ATN (IN HOLDING AREA)
2737  FC A7 25          CM    X25A7 IF ARG IS MINUS PUT 1813 ON STACK TO INVERT RESULT WHEN DONE   *%   ATN
273A  FC 13 18          CM    X1813 IF ARG IS MINUS, INVERT ARG (ARG FOR POLYNOMIAL MUST BE POSITIVE)
2730  3A 15 04          LDA   X0415 } TEST EXPONENT BYTE OF ARG IN HOLDING AREA            :
2740  FE 81             CPI   H'81' }
2742  DA 51 27          JC    X2751 JMP IF ARGUMENT MAGNITUDE IS LESS THAN 1             ZQ'
2745  01 00 81          LXI   B,X8100 } LOAD BCDE WITH SNG-PREC CONSTANT = 1
2748  51                MOV   D,C     }                                                  Q
2749  59                MOV   E,C     }                                                  Y
274A  CD 2E 1A          CALL  X1A2E PERFORM SNG-PREC DIVIDE, COMPUTE 1/ARG               M.
274D  21 96 18          LXI   H,X1896 PUT 1896 ON STACK - WILL COMPUTE (π/2 - RESULT) see 2757   !
2750  E5                PUSH  H
2751  21 5B 27 X2751    LXI   H,X275B LOAD HL WITH ADDR OF ITERATION CTR OF ATN POLYNOMIAL TABLE  !('
2754  CD 58 26          CALL  X2658 EVALUATE ATN POLYNOMIAL                              MX&
2757  21 04 27          LXI   H,X2704 LOAD HL WITH ADDR OF CONSTANT π/2 - used if 274D was executed  ! '
275A  C9                RET   END OF ATN - may do 1 or 2 other routines (see 2737 + 274D)   I
2758  09       X275B    DAD   B CONSTANT TO INDICATE 9 SINGLE-PRECISION FLOATING-POINT CONSTANTS IN THIS TABLE
275C  4A                MOV   C,D  }  SINGLE PRECISION CONSTANT                          J
275D  97                RST   2    }  = .00286623                                        W
275E  3B                DCX   SP   }                                                     ;
275F  78                MOV   A,B  }
2760  02                STAX  B    }  SINGLE PRECISION CONSTANT
2761  6E                MOV   L,M  }  = -.0161657
2762  84                ADD   H    }
2763  7B                MOV   A,E  }
2764  FE C1             CPI   H'C1' } SINGLE PRECISION CONSTANT                          A
2766  2F                CMA         } = .0429096                                         /
2767  7C                MOV   A,H   }
2768  74                MOV   M,H   } SINGLE PRECISION CONSTANT
2769  31 9A 7D          LXI   SP,X7D9A } = -.0752896                                     1
276C  84                ADD   H    }  SINGLE PRECISION CONSTANT
276D  3D                DCR   A    }  = .106563                                          =
276E  5A                MOV   E,D  }                                                     Z
276F  7D                MOV   A,L  }
2770  C8                RZ         }  SINGLE PRECISION CONSTANT                          H
2771  7F                MOV   A,A  }  = -.142089
2772  91                SUB   C    }
2773  7E                MOV   A,M  }
2774  E4 B0 4C          CPO   X4CB0 } SINGLE PRECISION CONSTANT                          'L
2777  7E                MOV   A,M   } = .199936
2778  6C                MOV   L,H  }  SINGLE PRECISION CONSTANT
2779  AA                XRA   D    }                                                     *
277A  AA                XRA   D    }  = -.333331                                         *
277B  7F                MOV   A,A  }
277C  00                NOP        }  SINGLE PRECISION CONSTANT
                                   }  = 1
```



if x is minus, invert arg + invert result

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

```
277D  00                NOP      } PART OF 9TH SINGLE
277E  00                NOP      } PRECISION CONSTANT = 1
277F  81                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 D0 03          SHLD X03D0    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 # + OUTPUT CONSOLE STATUS PORT # (IF INPUT + OUTPUT PORT THE SAME C = -1)
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-
                                                                } LOADED INTO LOC 2FFF (12K)
27A5  1F                RAR           ROTATE D8 INTO CARRY BIT  } 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)
27AC  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 1ØH 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 OF 2
27B4  DB FF             IN   H'FF'    READ HI-8 SWITCHES ON FRONT PANEL
27B6  17                RAL          } TEST BIT CORRESPONDING TO SW14   } SW14 UP
27B7  17                RAL          }                                  } SIO A,B,C (REV Ø)
27B8  06 20             MVI  B,A'     JZ
27BA  11 D2 CA X27BA    LXI  D,XCAD2
27BD  D8                RC            JMP IF SW14 UP TO 27FD
27BE  17                RAL           TEST BIT CORRESPONDING TO SW13    } SW13 UP
27BF  43                MOV  B,E                                        } 88-PIO
27C0  1D                DCR  E
27C1  D8                RC            JMP IF SW13 UP TO 27FD
27C2  17                RAL           TEST BIT CORRESPONDING TO SW12    } SW12 UP
                                                                        } 4PIO
27C3  DA DE 27          JC   X27DE    JMP IF SW12 UP
27C6  43                MOV  B,E      JNZ                               } SIO A,B,C (not REV Ø)
27C7  11 80 C2          LXI  D,XC280
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    } SW11 UP
                                                                        } 2SIO
27CD  3E 03             MVI  A,H'03'
27CF  CD FA 27          CALL X27FA
27D2  3D                DCR  A
27D3  8F                ADC  A
27D4  87                ADD  A
27D5  87                ADD  A
27D6  3C                INR  A
27D7  CD FA 27          CALL X27FA    A1Ø UP, SEND 15H TO PORT 1ØH; A1Ø DOWN, SEND 11H TO PORT 1ØH
27DA  37                STC           FORCE CARRY SET FOR IMPLIED JMP
27DB  C3 BA 27          JMP  X27BA
27DE  AF       X27DE    XRA  A        CLEAR ACC                         } 4PIO
```

INITIALIZATION OF BASIC

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
SIO A,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  49                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, 27FB 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 F0 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  81                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  }                                         "H    6399
2831  AF                XRA  A    } CLEAR PRINT SUPPRESSION FLAG              /     6400
2832  32 A2 03          STA  X03A2 }                                          2"    6401
2835  CD 99 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 A9 03          SHLD X03A9  }                                         "(    6405
2841  21 C0 29          LXI  H,X29C0 LOAD HL WITH ADDR OF CHAR STRING "MEMORY SIZE"(K) 6406  SET AMOUNT OF MEMORY
2844  CD 41 12          CALL X1241  PRINT CHAR STRING                         MA    6407
2847  CD E6 05          CALL X05E6  PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE, STORE M 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 AA 27          JZ   X27AA } 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 #     8 (   6413
2854  21 C7 29          LXI  H,X29C7 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              DCR  A     } STORE FF IN MEMORY LOC                                       =
2860 77              MOV  M,A   }
2861 BE              CMP  M     } TEST IF LOC. IS UNPROTECTED RAM                              >
2862 CA 57 28        JZ   X2857 TEST NEXT ADDRESS                                              JW(
2865 C3 74 28        JMP  X2874 JMP, END OF USABLE SEQUENTIAL RAM HAS BEEN FOUND              8 (
2868 21 5A 03 X2868  LXI  H,X035A LOAD HL WITH ADDRESS OF LINE BUFR                           !Z
286B CD F3 08        CALL X08F3 ASSEMBLE GROUP OF DIGITS INTO 2-BYTE VALUE IN DE              H    NOT TO EXCEED 65529
286E B7              ORA  A     } IF CHAR IN LINE BUFR AFTER LAST DIGIT IS NOT A NULL         7
286F C2 A9 04        JNZ  X04A9 } OR COLON, PRINT ERROR MESSAGE "SYNTAX ERROR" + RESTART       B)   INITIALIZATION
2872 EB              XCHG       PUT # FOR LAST ADDRESS  AVAILABLE TO BASIC IN HL
2873 2B              DCX  H     } ADJUST #                                                     +
2874 2B      X2874   DCX  H     }                                                              +
2875 E5              PUSH H PUT ADDR OF LAST BYTE AVAILABLE TO BASIC ON STACK
```

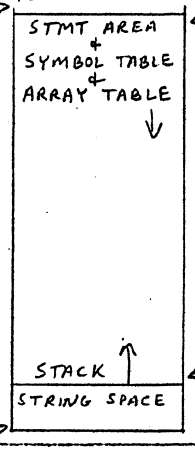
SET TERMINAL WIDTH

```
2876 21 7E 29 X2876  LXI  H,X297E LOAD HL WITH ADDR OF CHAR STRING "TERMINAL WIDTH"            ! )
2879 CD 41 12        CALL X1241 PRINT CHAR STRING                                              MA
287C CD E6 05        CALL X05E6 PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE INTO LINE     VE
287F D7              RST  2 SCAN OVER LEADING "␣"'s                                            H BUFR
2880 B7              ORA  A     } IF A RETURN WAS TYPED AFTER "?␣", LEAVE TERMINAL             7
2881 CA AB 28        JZ   X28AB } SET TO 72 DECIMAL FOR TTY ('48H)                             J+(
2884 21 5A 03        LXI  H,X035A LOAD HL WITH ADDR OF LINE BUFR                               !Z
2887 CD F3 08        CALL X08F3 ASSEMBLE GROUP OF DIGITS INTO A 2 BYTE VALUE                   M
288A 7A              MOV  A,D   } TEST HI-ORDER BYTE OF VALUE
288B B7              ORA  A     } IF NOT Ø, START OVER, TERMINAL WIDTH MUST BE < 256           7
288C C2 76 28        JNZ  X2876                                                                B (
288F 7B              MOV  A,E   } TEST LO-ORDER BYTE OF VALUE
2890 FE 10           CPI  H'10' }
2892 DA 76 28        JC   X2876 IF <16, START OVER AGAIN        16 ≤ WIDTH < 255               Z (
2895 32 09 06        STA  X0609 STORE TERMINAL WIDTH IN ROUTINE TO PRINT ASCII CHAR            2Y
2898 32 36 08        STA  X0836   "     "       "    " NULL FUNCTION ROUTINE                  26
289B 32 81 0A        STA  X0A81   "     "       "    " PRINT ROUTINE                          2
289E D6 0E    X289E  SUI  H'0E' } COMPUTE WIDTH - [MOD14(WIDTH) + 14]                          V
28A0 D2 9E 28        JNC  X289E   THIS # WILL IDENTIFY WHEN NEXT 14 CHAR                       R (
28A3 C6 1C           ADI  H'1C'   UNFORMATTED FIELD WILL NOT FIT ON A LINE                     F
28A5 2F              CMA          (SEE ØABØ IN PRINT ROUTINE)                                  /
28A6 3C              INR  A                                                                    <
28A7 83              ADD  E
28A8 32 94 0A        STA  X0A94 STORE VALUE IN PRINT ROUTINE                                   24
28AB 11 9D FF X28AB  LXI  D,XFF9D LOAD DE WITH 16 BIT VALUE = -99 DECIMAL (AMOUNT OF SPACE FOR STRINGS - DEFAULT VALUE)
28AE E1              POP  H FETCH ADDR FROM STACK OF LAST USABLE BYTE FOR BASIC
28AF 22 A6 03        SHLD X03A6 } SET START ADDR OF STRING SPACE TO HIGHEST AVAIL. ADDR        &S
28B2 22 CB 03        SHLD X03CB } SET END OF STRING SPACE TO SAME ADDR                         "K
28B5 19              DAD  D COMPUTE (LAST ADDR) - 100 = ADDR WHERE STRING SPACE ENDS
28B6 D2 9E 04        JNC  X049E IF (LAST ADDR) WAS 98 OR LESS PRINT MESSAGE "OUT OF MEMORY" + RESTART INITIALIZATION
28B9 2B              DCX  H ADJUST ADDR TO 1 LESS THAN LAST STRING AREA ADDR                   +
28BA E5              PUSH H SAVE ADDR ON STACK
```

SAVE OR ELIMINATE TRIG FUNCTIONS

```
28BB 21 2F 29 X28BB  LXI  H,X292F LOAD HL WITH ADDR OF CHAR STRING "WANT SIN-....-ATN?"        /)
28BE CD 41 12        CALL X1241 PRINT CHAR STRING                                              MA
28C1 CD E6 05        CALL X05E6 PRINT "?␣", THEN FETCH CHAR'S FROM INPUT CONSOLE INTO LINE BUFR  M LINE / H BUFR
28C4 D7              RST  2     SCAN OVER LEADING "␣"'s
28C5 FE 59           CPI  A'Y'  TEST 1ST NON-SPACE CHAR IN LINE BUFR                           Y
28C7 11 80 27        LXI  D,X2780 LOAD DE WITH ADDR OF BYTE AFTER ATN FUNCTION ROUTINE
28CA CA F1 28        JZ   X28F1 JMP IF 1ST CHAR WAS A 'Y', RETAIN ALL TRIG FCNS                J (
28CD FE 41           CPI  A'A'  } IF 1ST CHAR WAS A 'A', DELETE ONLY THE ATN FUNCTION           A
28CF CA 07 28        JZ   X2807 }                                                              JW(
28D2 FE 4E           CPI  A'N'  } IF 1ST CHAR WAS NOT A 'N', THEN ANSWER TO QUESTION           N
28D4 C2 BB 28        JNZ  X28BB } IS WRONG, REPEAT QUESTION                                    B:(
28D7 21 EE 08 X28D7  LXI  H,X08EE LOAD HL WITH ADDR OF ROUTINE TO PRINT ERROR MESSAGE "ILLEGAL FUNCTION CALL"   !
28DA 11 36 27        LXI  D,X2736 LOAD DE WITH ADDR OF BYTE AFTER SIN-COS-TAN FUNCTIONS        6'
28DD 22 57 00        SHLD X0057 DELETE REFERENCE TO ATN FUNCTION SUBROUTINE ADDR - CHANGE ADDR TO ERROR MESSAGE ROUTINE   "H
28E0 FE 41           CPI  A'A'  TEST 1ST CHAR IN LINE BUFR AGAIN                               A
```

```
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              X28F1 XCHG    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 Ø30D              "}        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
2909  78              MOV   A,E  }                                                             6498
290A  95              SUB   L    } COMPUTE DIFFERENCE BETWEEN DE & HL                          6499
290B  6F              MOV   L,A  }                                                             6500
290C  7A              MOV   A,D  }   (DE) - (HL) = # OF BYTES AVAILABLE                        6501
290D  9C              SBB   H    }                                                             6502
290E  67              MOV   H,A  }   RESULT IN HL                                              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 80 29        LXI   H,X2980 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 INPT.     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 ØØØ2, JMP TO COMMAND MODE PROCESSING      !^      6513
292B  22 02 00        SHLD  X0002 } ROUTINE, NOT INITIALIZATION                        "       6514
292E  E9              PCHL  JMP TO ADDR Ø4DE -- 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  0C        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
```



```
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
O 6584
C 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              D   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
298E  42              MOV  B,D              B   6608  "␣ BYTES FREE" CRLF CRLF
298F  59              MOV  E,C              Y   6609
2990  54              MOV  D,H              T   6610  "ALTAIR BASIC VERSION 3.2" CRLF
2991  45              MOV  B,L              E   6611
2992  53              MOV  D,E              S   6612  "[EXTENDED VERSION]" CRLF
2993  20              DATA A' '                 6613
2994  46              MOV  B,M              F   6614  STRING TERMINATED BY A NULL
2995  52              MOV  D,D              R   6615
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 32           MVI  L,H'32'          .2  6644
29B4  0D              DCR  C                    6645
29B5  0A              LDAX B                    6646
29B6  5B              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              D   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
29C4  49          MOV  C,C       I
29C5  4F          MOV  C,A       O
29C6  4E          MOV  C,M       N
29C7  C0          DATA H'D0'     )
29C8  0D          DCR  C
29C9  0A          LDAX B
29CA  00          NOP
-------------------------------------------------------
29CB  40   X29CB  MOV  C,L       M   CHAR STRING
29CC  45          MOV  B,L       E   "MEMORY SIZE"
29CD  4D          MOV  C,L       M
29CE  4F          MOV  C,A       O   STRING TERMINATED BY A NULL
29CF  52          MOV  D,D       R
29D0  59          MOV  E,C       Y
29D1  20          DATA A' '
29D2  53          MOV  D,E       S
29D3  49          MOV  C,C       I
29D4  5A          MOV  E,D       Z
29D5  C5          PUSH B         E
29D6  00          NOP
-------------------------------------------------------
29D7  00   X29D7  NOP
29D8  00          NOP
29D9  00          NOP
29DA  00          NOP
29DB  00          NOP
29DC  00          NOP
29DD  00          NOP
29DE  00          NOP
29DF  00          NOP
```