Nola's SFUN EF STR EA EVAL ES ARRAY NONCPL

```
 .INSERT A:MAC.ASM
@
 .INSERT A:S.ASM
@.REMARK /
@                        *
@                      * * *
@                     ***
@                    *******
@                         *
@                        *
@                       *
@                      *
@                     *
@                    *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V2.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                    (C) 1978
@
@/
 .INSERT A:ZRAM.ASM
 .LINK
 .IDENT  SFUN
 .PREL
 ;+
 ; INTERNALS
 ;-
 .INTERN FIND                ;FIND STRING
 .INTERN LPAD                ;PAD ON LEFT
 .INTERN SUBSTR              ;SUBSTRING
 ;+
 ; EXTERNALS
 ;-
 .EXTERN ALSTS               ;ALLOCATE STRING
 .EXTERN ARG                 ;ARGUMENT PARSER
 .EXTERN DIV16               ;DIVIDE HL BY 16
 .EXTERN ENDSTR              ;CLEAN UP FOR STRING
 .EXTERN ERRPGM              ;ERRORS
 .EXTERN GETOPND             ;GET OPERAND
 .EXTERN INC5IY              ;BUMP IY TO NEXT ARG
 .EXTERN RETNONE             ;RETURN POINT
 .EXTERN SFREE               ;FREE STRING
 ;++++
 ;
 ; LPAD
 ; PAD STRING ON THE LEFT WITH GIVEN CHARACTER SO IT FITS

 ; IN A SPECIFIED FIELD
 ;
 ; SYNTAX:
 ;       LPAD <STRING>,<CHAR>,<FIELD>
 ;    <STRING> - STRING TO PAD
```

```
                        ;    <CHAR> - STRING WHOSE FIRST CHARACTER WILL BE USED A
                        S
                        ;        THE PAD CHARACTER
                        ;    <FIELD> - INTEGER GIVING THE LENGTH DESIRED FOR THE
                        FIELD
                        ;
                        ; RETURNS:
                        ;    STRING RETURNED WILL BE <FIELD> CHARACTERS LONG. IF
                        ORIGINAL
                        ;    <STRING> WAS TOO SHORT, IT IS PADDED ON THE LEFT WIT
                        H THE FIRST
                        ;    CHARACTER OF THE STRING <CHAR> UNTIL IT FITS IN THE
                        FIELD
                        ;    IF IT WAS TOO LONG, IT IS TRUNCATED ON THE LEFT
                        ;
                        ; CALLS:
                        ;    ARG - ARGUMENT PARSER
                        ;    GETSTO - GET STRING OPERAND
                        ;    CHKINT - GET INTEGER OPERAND
                        ;    SLEN - STRING LENGTH FINDER
                        ;    SFREE - FREE STRING
                        ;    DIV16 - DIVIDE BY 16
                        ;    ALSTS - ALLOCATE STRING
                        ;    ENDSTR - CLEAN UP STRING
                        ;
                        ;----
                                M.CMD[LPAD,..LPX,0,AFIX[L],..LPARG,..LPAD][
0000'   18              +LPAD:  .BYTE $CMDADR
0001'   07              +       .BYTE (..LPX-LPAD)/16+1
0002'   00              +       .BYTE 0
0003'   639C            +       .WORD AFIX[L][("L"-101Q)*16+FSTINT]
0005'   000E'          +       .WORD ..LPAD
0007'   0011'          +       .WORD ..LPARG
0009'   4C50414400      +       .ASCIZ /LPAD/
                        +]
000E'   CD 0000:05      ..LPAD: CALL    ARG
0011'   08080400        ..LPARG:.BYTE   $STRADR,$STRADR,$IVAL,$TAF
0015'   E5                      PUSH    H
                        ;+
                        ; THE FIRST ARGUMENT IS THE STRING TO PAD
                        ; IT MUST BE NON-NULL OR IT IS AN ERROR
                        ;-
0016'   CD 006E'                CALL    GETSTO          ;GET 1ST STRING
0019'   E5                      PUSH    H               ;HL->TOP OF STRING
001A'   D5                      PUSH    D               ;DE->1ST CHAR OF STRING
001B'   CD 0095'                CALL    SLEN            ;B=LENGTH OF STRING
                        ;+
                        ; SP -> TOP OF SOURCE
                        ;       1ST CHAR OF SOURCE
                        ; B = LENGTH OF SOURCE
                        ; THE SECOND OPERAND IS THE PAD CHARACTER. ONLY
                        ; THE FIRST CHARACTER IS USED. AN ATTEMPT IS MADE
                        ; TO GARBAGE COLLECT THE PAD STRING
                        ;-
001E'   CD 006E'                CALL    GETSTO          ;GET PAD STRING
```

```
0021'  1A                    LDAX    D               ;A = PAD CHAR
0022'  4F                    MOV     C,A             ;C = PAD CHARACTER
0023'  CD 0000:0C            CALL    SFREE           ;FREE IT
                       ;+
                       ; SP -> TOP OF SOURCE
                       ;       1ST CHAR OF SOURCE
                       ; B = LENGTH OF SOURCE
                       ; C = PAD CHARACTER
                       ; THE LAST OPERAND IS AN INTEGER GIVING THE SIZE OF
                       ; THE FIELD NEEDED. IF THE STRING TO BE PADDED HAS
                       ; FEWER CHARS THAN THE FIELD LENGTH, IT IS PADDED
                       ; ON THE LEFT WITH THE PAD CHARACTER. IF IT HAS MORE,
                       ; IT IS TRUNCATED ON THE LEFT TO FIT IN THE FIELD
                       ;-
0026'  CD 0083'              CALL    CHKINT          ;GET FIELD LENGTH
0029'  7B                    MOV     A,E             ;GET IN A
002A'  E1                    POP     H               ;HL->1ST SOURCE CHAR
002B'  90                    SUB     B               ;A=# TO PAD
002C'  FA 0038'              JM      ..L1            ;<0? MUST TRUNCATE
002F'  2016                  JRNZ    ..L5            ;>0? MUST PAD
                       ;+
                       ; SP -> TOP OF SOURCE
                       ; THE STRING IS EXACTLY THE FIELD LENGTH. NOTHING
                       ; NEED BE DONE TO IT, IT IS RETURNED ASIS
                       ;-
0031'  D1              ..LEX:  POP     D
0032'  3E08            ..EXIT: MVI     A,$STRADR       ;TYPE STRING
0034'  E1                      POP     H               ;RESTORE ARG PTR
0035'  C3 0000:0B             JMP     RETNONE
                       ;+
                       ; SP -> TOP OF SOURCE
                       ;       1ST CHAR OF SOURCE
                       ; A = # OF CHARS TO TRUNCATE
                       ; THE STRING MUST BE TRUNCATED ON THE LEFT, IT IS
                       ; TOO SMALL TO FIT IN THE FIELD. THE POINTER
                       ; (HL) IS ADVANCED TO THE FIRST CHAR TO GO IN
                       ; THE FIELD AND THEN ALL CHARACTERS ARE MOVED
                       ; OVER (SHUFFLED)
                       ;-
0038'  E5              ..L1:   PUSH    H               ;SAVE CHAR PTR
0039'  23              ..L2:   INX     H               ;SKIP CHARS TO
003A'  3C                      INR     A               ;TRUNCATE STRING
003B'  20FC                    JRNZ    ..L2
003D'  D1                      POP     D               ;DE->1ST CHAR POS
003E'  7E              ..L3:   MOV     A,M             ;GET NEXT CHAR
003F'  12                      STAX    D               ;SAVE IN STRING
0040'  23                      INX     H               ;BUMP PTRS
0041'  13                      INX     D
                               TST     A               ;UNTIL NULL[
0042'  B7              +       ORA     A]
0043'  20F9                    JRNZ    ..L3
0045'  18EA                    JMPR    ..LEX
                       ;+
                       ; THE STRING MUST BE PADDED ON THE LEFT WITH THE
                       ; PAD CHARACTER. A NEW STRING IS CREATED LARGE
```

```
                        ; ENOUGH TO HOLD THE NUMBER OF CHARACTERS IN THE
                        ; FIELD
                        ;-
0047'  47               ..L5:   MOV     B,A             ;SAVE # TO PAD
0048'  E5                       PUSH    H               ;HL->1ST CHAR
0049'  C5                       PUSH    B
004A'  21 000B                  LXI     H,$SASCII+1     ;HEADER PLUS NULL
004D'  19                       DAD     D               ;HL=# OF CHARS
004E'  CD 0000:06               CALL    DIV16           ;GET # BLOCKS
0051'  CD 0000:04               CALL    ALSTS           ;ALLOCATE STRING
                        ;+
                        ; SP -> TOP OF SOURCE
                        ;       PAD LENGTH, PAD CHAR
                        ;       1ST CHAR OF SOURCE
                        ; DE -> 1ST CHAR OF NEW STRING
                        ; STRING IS CREATED, IT IS PADDED ON THE LEFT WITH
                        ; THE APPROPRIATE NUMBER OF PAD CHARACTERS
                        ; DE -> 1ST BYTE OF NEW STRING
                        ;-
0054'  C1                       POP     B               ;B=PAD NUM,C=PAD CHAR
0055'  E3                       XTHL                    ;HL->1ST CHAR OF SOURCE
0056'  79                       MOV     A,C             ;A = PAD CHAR
0057'  12               ..L6:   STAX    D               ;PAD ONE
0058'  13                       INX     D               ;BUMP PTR
0059'  05                       DCR     B               ;FOR B CHARS
005A'  20FB                     JRNZ    ..L6
                        ;+
                        ; HL -> 1ST CHAR IN SOURCE STRING
                        ; DE -> NEXT BYTE IN DESTINATION
                        ; THE REST OF THE OLD STRING IS COPIED OVER UNTIL
                        ; THE NULL AND THE OLD STRING IS GARBAGE COLLECTED
                        ;-
005C'  7E               ..L7:   MOV     A,M             ;COPY A CHAR
005D'  12                       STAX    D
005E'  23                       INX     H               ;BUMP PTRS
005F'  13                       INX     D
                                TST     A               ;IS NULL?[
0060'  B7               +       ORA     A]
0061'  20F9                     JRNZ    ..L7            ;EXIT
0063'  1B                       DCX     D               ;AFTER NULL
0064'  E1                       POP     H               ;HL->TOP OF NEW
0065'  CD 0000:07               CALL    ENDSTR          ;CLEAN UP
0068'  E1                       POP     H               ;FREE OLD STRING
0069'  CD 0000:0C               CALL    SFREE
006C'  18C4                     JMPR    ..EXIT          ;AND EXIT
006E'                   ..LPX:
                        ;++++
                        ;
                        ; GETSTO
                        ; GET STRING OPERAND, FIND 1ST CHARACTER, ERROR IF NULL
                        ;
                        ; NEEDS:
                        ;   IY-> POSSIBLE STRING OPERAND
                        ;
                        ; RETURNS:
```

```
                        ;  IY -> NEXT OPERAND
                        ;  HL -> TOP OF STRING
                        ;  DE -> 1ST CHAR
                        ;
                        ; ERRORS:
                        ;   ER.NULS - ERROR IF NULL
                        ;
                        ; CALLS:
                        ;   GETOPND - GET THE OPERAND
                        ;   INC5IY - BUMP IY TO NEXT OPERAND
                        ;
                        ;-----
006E'  CD 0000:09   GETSTO: CALL    GETOPND         ;GET STRING
                            TSTD    D               ;IS NULL?[
0071'  7A           +       MOV     A,D
0072'  B3           +       ORA     D               +1]
                            ZERROR  ER.NULS         ;YES? ERROR[
0073'  2004         +       JRNZ    ..0001
0075'  CD 0000:08   +       CALL    ERRPGM
0078'  39           +       .BYTE   ER.NULS
0079'               +..0001:]
0079'  D5                   PUSH    D               ;SAVE TOP PTR
007A'  21 000A               LXI     H,$SASCII       ;HL=OFFSET OF 1ST
007D'  19                    DAD     D               ;HL->1ST CHAR
007E'  EB                    XCHG                    ;HL->TOP,DE->1ST CHAR
007F'  E1                    POP     H
0080'  C3 0000:0A            JMP     INC5IY          ;IY->NEXT ARG
                        ;++++
                        ;
                        ; CHKINT
                        ; GET INTEGER OPERAND, ERROR IF NEGATIVE, CHECK IF ZERO
                        ;
                        ; NEEDS:
                        ;   IY -> NEXT OPERAND
                        ;
                        ; RETURNS:
                        ;   DE = INTEGER OPERAND (NON-NEGATIVE)
                        ;   IY -> AFTER INTEGER OPERAND
                        ;   Z BIT SET IF DE=0, ELSE CLEAR
                        ;
                        ; ERRORS:
                        ;   ER.NEG - NEGATIVE ARGUMENT NOT ALLOWED HERE
                        ;
                        ; CALLS:
                        ;   GETOPND - GET OPERAND
                        ;   INC5IY - POINT AT NEXT OPERAND
                        ;
                        ;-----
0083'  CD 0000:09   CHKINT: CALL    GETOPND         ;DE=OPERAND
0086'  CD 0000:0A           CALL    INC5IY          ;NEXT ONE
                            TST     D               ;IS IT NEGATIVE?[
0089'  7A           +.IFN D         -A, [   MOV     A,D             ]
008A'  B7           +       ORA     A]
008B'  FA 0091'             JM      ..ERR           ;YES? ERROR
                            TSTD    D               ;IS IT ZERO?[
```

```
008E'  7A            +          MOV     A,D
008F'  B3            +          ORA     D                +1]
0090'  C9                       RET
0091'                ..ERR:     ERROR   ER.NEG           ;NEGATIVE ARG[
0091'  CD 0000:08    +          CALL ERRPGM
0094'  3A            +          .BYTE ER.NEG
                     +]
                      ;++++
                      ;
                      ; SLEN
                      ; FIND LENGTH OF STRING
                      ;
                      ; NEEDS:
                      ;   DE -> 1ST CHAR OF STRING
                      ;
                      ; RETURNS:
                      ;   DE -> AFTER NULL
                      ;   B = # OF CHAR IN STRING (NOT INCLUDING NULL)
                      ;
                      ;-----
0095'                 SLEN:      CLR     B                ;0 CHARS[
0095'  0600          +.IFN B               -A, [   MVI     B                ,0][
                      XRA     A]]
0097'  1A             ..S1:      LDAX    D                ;GET A CHAR
                                 TST     A                ;IS NULL?[
0098'  B7            +          ORA     A]
0099'  C8                       RZ                       ;RETURN THEN
009A'  13                       INX     D                ;BUMP PTR
009B'  04                       INR     B                ;AND COUNTER
009C'  F2 0097'                 JP      ..S1             ;UNTIL OVERFLOW
                                ERROR   ER.OFLO          ;STRING TOO LONG[
009F'  CD 0000:08    +          CALL ERRPGM
00A2'  17            +          .BYTE ER.OFLO
                     +]
                      ;++++
                      ;
                      ; SUBSTR
                      ; FINDS THE SPECIFIED SUBSTRING
                      ;
                      ; SYNTAX:
                      ;       SUBSTR STRING,START,LEN
                      ;   <STRING> - STRING TO TAKE SUBSTRING OUT OF
                      ;   <START> - INTEGER SPECIFYING THE CHARACTER TO START
                      ;       THE SUBSTRING AT (0 BASED)
                      ;   <LEN> - NUMBER OF CHARACTERS IN SUBSTRING
                      ;
                      ; RETURNS:
                      ;   SUBSTRING STARTING AT THE SPECIFIED CHARACTER WHICH
                      IS
                      ;   <LEN> CHARACTERS LONG
                      ;
                      ; CALLS:
                      ;   GETSTO - GET STRING OPERAND
                      ;   CHKINT - GET INTEGER OPERAND
                      ;   SFREE - FREE STRING
```

```
                        ;   ALSTS - ALLOCATE STRING
                        ;   ENDSTR - CLEAN UP AFTER STRING
                        ;
                        ;----
                                M.CMD[SUBSTR,..SBX,0,AFIX[S],..SARG,..SUB][
00A3'   18              +SUBSTR: .BYTE $CMDADR
00A4'   06              +        .BYTE (..SBX-SUBSTR)/16+1
00A5'   00              +        .BYTE 0
00A6'   640C            +        .WORD AFIX[S][("S"-101Q)*16+FSTINT]
00A8'   00B3'           +        .WORD ..SUB
00AA'   00B6'           +        .WORD ..SARG
00AC'   535542535452+   +        .ASCIZ /SUBSTR/
                        +]
00B3'   CD 0000:05      ..SUB:   CALL    ARG
00B6'   08040400        ..SARG:  .BYTE   $STRADR,$IVAL,$IVAL,$TAF
                        ;+
                        ; GET SOURCE OPERAND AND SAVE POINTERS
                        ;-
00BA'   CD 006E'                 CALL    GETSTO          ;GET SOURCE
00BD'   E5                       PUSH    H               ;HL->TOP OF STRING
00BE'   D5                       PUSH    D               ;DE->1ST CHAR
                        ;+
                        ; GET FIRST CHARACTER INDEX AND SKIP TO THAT CHARACTER
                        ; IN THE SOURCE STRING
                        ;-
00BF'   CD 0083'                 CALL    CHKINT          ;GET 1ST CHAR OFFSET
00C2'   E1                       POP     H               ;HL->1ST CHAR OF SOURCE
00C3'   280A            ..S1:    JRZ     ..S2            ;YES? GO ON
                                 TST     M               ;RAN OUT OF CHARS?[
00C5'   7E              +.IFN M          -A, [   MOV     A,M             ]
00C6'   B7              +        ORA     A]
00C7'   282F                     JRZ     ..NULL          ;RETURN NULL STRING
00C9'   23                       INX     H               ;SKIP ONE MORE
00CA'   1B                       DCX     D               ;DEC COUNTER
                                 TSTD    D               ;CHECK IF 0[
00CB'   7A              +        MOV     A,D
00CC'   B3              +        ORA     D               +1]
00CD'   18F4                     JMPR    ..S1
                        ;+
                        ; ALLOCATE ENOUGH MEMORY FOR THE SUBSTRING
                        ; THE CHARACTER COUNT MUST BE CONVERTED TO A STRING BLOC
                        K
                        ; COUNT AND USED FOR ALSTS
                        ; IF THE LENGTH IS 0, IT MEANS THE REST OF THE SOURCE
                        ;-
00CF'   CD 0083'        ..S2:    CALL    CHKINT          ;GET SUBSTR LENGTH
00D2'   E5                       PUSH    H               ;SAVE STRING PTR
00D3'   D5                       PUSH    D               ;SAVE SUBSTR LEN
00D4'   3EFF                     MVI     A,-1            ;IN CASE DE=0
00D6'   2808                     JRZ     ..S3            ;0? UNKNOWN SIZE
00D8'   21 000B                  LXI     H,$SASCII+1     ;HEADER AND NULL
00DB'   19                       DAD     D               ;HL=BYTE SIZE
00DC'   CD 0000:06               CALL    DIV16           ;HL=BLOCK SIZE
00DF'   7D                       MOV     A,L             ;A=BLOCK SIZE
00E0'   CD 0000:04      ..S3:    CALL    ALSTS           ;ALLOCATE IT
```

```
00E3'  C1                      POP     B               ;BC=LENGTH
00E4'  E3                      XTHL                    ;SAVE TOP PTR
                       ;+
                       ; THE SOURCE STRING IS COPIED INTO THE NEW DESTINATION
                       ; STRING UNTIL WE RUN OUT OF CHARS OR UNTIL BC=0
                       ; THEN WE CLEAN UP AND EXIT
                       ;-
00E5'                  ..S4:   TST     M               ;SOURCE CHAR NULL?[
00E5'  7E              +.IFN M         -A, [   MOV     A,M             ]
00E6'  B7              +               ORA     A]
00E7'  2805                    JRZ     ..S5            ;ALL DUN
00E9'  EDA0                    LDI                     ;COPY A CHAR
00EB'  EA 00E5'                JPE     ..S4            ;GO AGAIN
00EE'  E1              ..S5:   POP     H               ;HL->TOP OF SUBSTR
00EF'  CD 0000:07              CALL    ENDSTR          ;CLEAN UP
00F2'  E1                      POP     H               ;HL->TOP OF SOURCE
00F3'  CD 0000:0C              CALL    SFREE           ;FREE IF NEED BE
00F6'  1804                    JMPR    ..EX
                       ;+
                       ; COME HERE WHEN SUBSTRING IS NULL
                       ;-
00F8'  D1              ..NULL: POP     D               ;DE->SOURCE
00F9'  11 0000                 LXI     D,0             ;RETURN NULL
00FC'  3E08            ..EX:   MVI     A,$STRADR       ;RETURN TYPE STRING
00FE'  C3 0000:0B              JMP     RETNONE
0101'                  ..SBX:
0101'  C9              FIND:   RET
                       .END
```

SFUN -
+++++ SYMBOL TABLE +++++

```
AASN    005F        ALSTS   0000:04 X   ARG     0000:05 X   ARGSTK  60CC
BACKGR  65CC        BLANK   0020        BOTRAM  6000        BOTTOM  64A9
CHARSL  65DD        CHKINT  0083'       CLEARS  60CA        CNTRL   000C
CNTRLC  65CD        CNTRLO  65D9        CNTRLU  0015        CNTRLZ  65E4
CPLARE  611C        CPLSIZ  0140        CR      000D        CSBLOK  668D
CSFLAG  6260        CURCX   6814        CURCY   6816        CURREN  64A5
C.CO    0011        C.C1    0012        C.CX    000B        C.CY    000D
C.DP    0013        C.ST    0002        C.X     0003        C.XF    000F
C.XS    0007        C.Y     0005        C.YF    0010        C.YS    0009
DDTON   65CE        DEVBL   6589        DEVCLO  6579        DEVCL1  657B
DEVCL2  657D        DEVCL3  657F        DEVCL4  6581        DEVCL5  6583
DEVCL6  6585        DEVCL7  6587        DEVFB   65CB        DEVHCB  658F
DEVMO   658B        DEVNM   65B7        DEVNT   658D        DEVTNA  65BB
DEVTNB  65BF        DEVTNC  65C3        DEVVA   65BD        DEVVAR  6579
DEVVB   65C1        DEVVBL  6591        DEVVC   65C5        DEVVD   65C9
DEVVN   65B9        DEVVS   65C7        DEVXCD  65B3        DEVYCD  65B5
DIV16   0000:06 X   DOLPLH  62E8        DOLPPT  62EA        DUMBST  6577
EDBCNT  64AD        EDCCNT  64A7        EDLONG  6812        EDMODE  681C
EDNAME  64A1        EDNCX   680C        EDNCY   680E        EDNEWS  64A5
EDOCX   6808        EDOCY   680A        EDPN    6806        EDPO    6804
EDPTRC  64AB        EDPTRL  64A9        EDSTR   681A        ENDSTR  0000:07 X
ERABIT  0002        ERRPGM  0000:08 X   ER.ARA  002F        ER.ARG  0034
ER.ASN  0015        ER.BOX  001A        ER.CHN  0002        ER.CMD  001F
ER.CNV  0016        ER.COR  001B        ER.CTL  0036        ER.DEL  0026
ER.DIM  0030        ER.DIV  0018        ER.DP   0037        ER.DSK  0019
ER.EDT  0035        ER.FMT  0038        ER.FNF  001C        ER.FOR  0028
ER.IMP  0003        ER.LAB  0025        ER.MAC  0022        ER.NAE  002D
ER.NAM  0029        ER.NEG  003A        ER.NOT  0023        ER.NUL  0039
ER.NUM  002B        ER.NXT  001D        ER.OFL  0017        ER.OPN  0014
ER.OVE  001E        ER.PAR  002A        ER.REN  002C        ER.RET  0024
ER.SEP  0021        ER.SNP  0031        ER.SPC  002E        ER.STK  0004
ER.SW   0032        ER.TER  0020        ER.UFL  0033        ER.UNF  0027
EXTDEL  002E        E.HVAL  0002        E.LVAL  0001        E.SIZ   0005
E.TYP   0000        E.VAL   0001        FCNTH   65D2        FCNTI   65D3
FCNTJ   65D4        FCNTK   65D5        FCNTL   65D6        FCNTV   65E0
FCNTY   65E3        FIND    0101'   I   FIRST   64A3        FLAGS   65CB
FOREGR  65DO        FRAGSI  0400        FREELS  65E5        FSTDOL  648C
FSTINT  62EC        FWDPTR  65E7        GETOPN  0000:09 X   GETSTO  006E'
HCAREA  6593        INC5IY  0000:0A X   INCRO   65E9        JUNK    6542
KBLOCK  65F1        KEYFLG  67FF        KEYPTK  6533        KEYTRK  67F7
LF      000A        LISTON  65E2        LPAD    0000'   I   MACSTU  6536
MACTOP  625E        MAXFRG  0040        MNMX    65EB        NBLKB   0000
NBLKM   0001        NEWBOT  6810        NL      000A        NSADDR  6802
NUMBUF  64A3        OLDCHR  64A0        OLDCUR  649F        OLDKEY  649D
OLDXY   65EF        ONEBUF  6729        OPRL    0014        OPRSP   655B
OPRSTK  6547        OPRSZ   655D        OUTOFF  62E7        O.CHAR  0000
O.PREC  0002        O.SIZ   0006        O.SUB   0004        O.TYP   0003
PCNT    655E        PIXVAL  65ED        POINTE  64A7        PONOFF  65DA
PRINTR  6540        RAMEND  7FFF        RAMSTR  6900        RANSHT  655F
RETNON  0000:0B X   RMDTMP  6538        RUBACK  0005        RUBOUT  007F
SAVESP  625C        SCNEED  0004        SCRWIN  67CD        SFREE   0000:0C X
SLEN    0095'       SOPRSP  653D        SOPRSZ  653F        STACK   60C8
STAKTO  6000        STRSIZ  6800        SUBSTR  00A3'   I   SUBSTU  6534
TAB     0009        TAPBUF  64B3        TAPCON  64AF        TAPPRO  64B1
TBFEND  6533        TEMPHD  60CA        TEMPS   62E5        TMPARG  67AD
```

SFUN -
+++++ SYMBOL TABLE +++++

TOP	6818	TTYBEG	6261	TTYEND	62E1	TTYINT	62E3
TTYPTR	62E1	TXTWIN	67E2	UARTFL	649C	USREND	65F1
V3PTR	6573	VDCHAR	649E	VDNLF	65CF	VIPLH	6545
VOICE0	6563	WRMODE	65EE	ZGIM2	0001	ZGREND	681D
$AADR	0010	$ADDRF	0007	$ADDRI	0005	$ADDRS	0009
$ANY	001A	$ANYNA	FFFC	$ANYVA	FFFE	$ARGPT	0011
$BGPTR	000F	$BNDL	0007	$CALLE	000D	$CMDAD	0018
$CPLBL	002A	$CSBLO	0028	$DATAP	0007	$DOLDE	0001
$DOLOO	0002	$DVAL	0000	$END	001C	$FADR	000E
$FLAGS	0002	$FORBL	0024	$FORPT	000B	$FVAL	0006
$GOSUB	001A	$IADR	000C	$INPBU	0018	$INPPT	0016
$IVAL	0004	$KEYBL	0026	$LENGT	0001	$LINPT	0005
$LOCPT	0009	$MIBBL	0022	$MIBEN	001B	$NAMAD	000A
$NAME	000A	$NASCI	0009	$NDEL	0080	$NLINK	0003
$NULL	0002	$NVALU	0005	$REPEA	001E	$RVSTU	0013
$SAME	0020	$SASCI	000A	$SLEN	0006	$STRAD	0008
$STRPT	0003	$TAF	0000	$TYPE	0000	$USE	0005
.BLNK.	0000:03 X	.DATA.	0000* X	.PROG.	0102' X		

```
 .INSERT A:MAC.ASM
@
 .INSERT A:S.ASM
@.REMARK /
@                        *
@                      * * *
@                       ***
@                     *******
@                        *
@                       *
@                      *
@                     *
@                    *
@                    *******
@
@      "WHEN YOU CARE ENOUGH TO PROGRAM
@                 THE VERY BEST"
@
@              ZGRASS V2.00000000
@BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
@                   (C) 1978
@
@/
 .INSERT A:NCUEQU.ASM
 .INSERT A:ZRAM.ASM
 .LINK
 .IDENT  EF
 .PREL
 ;+
 ; INTERNALS
 ;-
 .INTERN ARCCOS             ;ARC COSINE FUNCTION
 .INTERN ARCSIN             ;ARC SINE FUNCTION
 .INTERN ARCTAN             ;ARC TANGENT FUNCTION
 .INTERN CNVFS              ;CONVERT FLOATING TO STRING
 .INTERN COSINE             ;COSINE FUNCTION
 .INTERN EXP                ;EXPONENT FUNCTION
 .INTERN FORMAT             ;FORMAT ASCII OUTPUT
 .INTERN GETNUM             ;PARSE FLOATING NUMBER
 .INTERN INT                ;INTEGER CONVERT
 .INTERN LN                 ;LOG NATURAL FUNCTION
 .INTERN LOG                ;LOG BASE 10 FUNCTION
 .INTERN PI                 ;RETURN VALUE OF PI
 .INTERN POWER              ;POWER FUNCTION
 .INTERN SINE               ;SINE FUNCTION
 .INTERN SQRT               ;SQUARE ROOT FUNCTION
 .INTERN TANGENT            ;TANGENT FUNCTION
 ;+
 ; EXTERNALS
 ;-
 .EXTERN ALSTS              ;ALLOCATE STRING
 .EXTERN ARG                ;PARSE ARGUMENTS
 .EXTERN CARTYP             ;FIND CHARACTER TYPE
 .EXTERN CVDSTR             ;CONVERT DOUBLE TO STRING
 .EXTERN CVFSTR             ;CONVERT FRACTION TO STRING
```

EF -

```
                        .EXTERN DONCU           ;DO NCU OPERATION
                        .EXTERN ENDSTR          ;CLEAN UP STRING
                        .EXTERN ERRPGM          ;ERROR TRAP
                        .EXTERN FPUSH           ;PUSH FLT NUMBER
                        .EXTERN GETMSB          ;GET MOST SIGNIF BYTE
                        .EXTERN GETOPND         ;GET OPERAND
                        .EXTERN INC5IY          ;POINT TO NEXT ARG
                        .EXTERN IPOP            ;POP INT FROM NCU STACK
                        .EXTERN IPSH0           ;PUSH 0 ONTO NCU STACK
                        .EXTERN IPUSH           ;PUSH INT ONTO NCU STACK
                        .EXTERN IYNCU           ;GET FROM IY TO NCU
                        .EXTERN LPAD            ;FOR LOG LINKAGE
                        .EXTERN NCUIY           ;GET FROM NCU TO IY
                        .EXTERN PLAY            ;FOR POWER LINK
                        .EXTERN POPOPND         ;POP OPERAND
                        .EXTERN PSHF10          ;PUSH FLOATING 10
                        .EXTERN PSHOPND         ;PUSH OPERAND
                        .EXTERN PUTOPND         ;PUT OPERAND
                        .EXTERN RETNONE         ;RETURN
                        .EXTERN SUBSTR          ;FOR SQRT LINK
                        ;++++
                        ;
                        ; CNVFS
                        ; CONVERT FLOATING POINT OPERAND TO STRING OPERAND
                        ;
                        ; NEEDS:
                        ;   IY-> FLOATING POINT OPERAND
                        ;
                        ; RETURNS:
                        ;   IY-> STRING OPERAND
                        ;
                        ; CALLS:
                        ;    ALSTS - ALLOCATE STRING
                        ;    ENDSTR - CLEAN UP AFTER STRING
                        ;    GETOPND, PUTOPND - FETCH, PUT OPERAND
                        ;    GETMSB - GET MOST SIGNIFICANT BYTE
                        ;    INTPT - CONVERT INTEGER PART
                        ;    FRACPT - CONVERT FRACTIONAL PART
                        ;    DONCU - DO NCU OPERATION
                        ;
                        ;----
0000'                   CNVFS:
0000'  C5                       PUSH    B               ;SAVE BC
0001'  3E01                     MVI     A,($ASCII+13)/16
0003'  CD 0000:04               CALL    ALSTS           ;ALLOCATE STRING
0006'  E5                       PUSH    H               ;SAVE PTR
0007'  EB                       XCHG                    ;HL->WHERE TO PUT DATA
0008'  CD 0000:0E               CALL    GETOPND         ;GET NUM ON NCU
000B'  CD 0041'                 CALL    MSGN            ;DO THE SIGN
000E'  CD 004F'                 CALL    NORMLIZ         ;NORMALIZE THE NUMBER
                        ;+
                        ; BC = BASE 10 EXPONENT
                        ; IF THE BASE 10 EXPONENT IS BETWEEN -3 AND 7 WE DO NOT
                        ; USE SCIENTIFIC NOTATION
                        ;-
```

```
EF      -
CNVFS - CONVERT FLOATING TO STRING


   0011'   79             ..CA:    MOV      A,C              ;A=BASE 10 EXP
   0012'   FE07                    CPI      6+1              ;EXP<7? NORMAL
   0014'   3810                    JRC      ..C2             ;EXP<7? NORMAL
   0016'   FEFD                    CPI      -3               ;EXP<-3? SCI NO
   0018'   300C                    JRNC     ..C2
   001A'   CD 00FE'        ..C1:    CALL     SCINO            ;SCIENTIFIC NOTATION
                           ;+
                           ; EXIT, CLEAN UP AFTER STRING AND PUSH IT
                           ;-
   001D'                   ..EX:    CLR      M                ;TRAILING NULL[
LL 001D'   3600           +.IFN M                -A?, ?[ MVI      M                ,0?][
                           XRA      A]]
   001F'   D1                      POP      D                ;HL->TOP OF STRING
   0020'   C1                      POP      B                ;RESTORE BC
   0021'   3E08                    MVI      A,$STRADR        ;TYPE STRING
   0023'   C3 0000:1A              JMP      PUTOPND          ;SAVE STRING
                           ;+
                           ; C = BASE 10 EXPONENT
                           ;    IF 0, .NNNNNN
                           ;    IF <0, .00NNNNNN
                           ;    ELSE, NN.NNNN
                           ;-
   0026'   CD 002B'        ..C2:    CALL     NUMIT            ;NORMAL NOTATION
   0029'   18F2                    JMPR     ..EX             ;AND EXIT
                           ;++++
                           ;
                           ; NUMIT
                           ; CONVERTS A FLOATING POINT NUMBER TO AN ASCII INTEGER
                           ; AND FRACTINAL PART
                           ;
                           ; NEEDS:
                           ;    BC = BASE 10 EXPONENT
                           ;    HL -> WHERE TO PUT FIRST DIGIT
                           ;    NCU STACK HAS NORMALIZED NUMBER
                           ;
                           ; RETURNS:
                           ;    HL -> AFTER LAST DIGIT
                           ;    BC DESTROYED
                           ;
                           ; CALLS:
                           ;    FLTOUT - GET 6 DIGITS
                           ;    FRAC, FRACPT - FRACTION PART
                           ;    INTPT - INTEGER PART
                           ;
                           ;----
   002B'   E5              NUMIT:   PUSH     H                ;SAVE OUR PTR
   002C'   21 64A3                 LXI      H,NUMBUF         ;HL->NUMBER BUFFER
   002F'   E5                      PUSH     H
   0030'   CD 00E7'                CALL     FLTOUT           ;GET 7 DIGITS
   0033'   D1                      POP      D                ;DE->NUMBER BUF
   0034'   E1                      POP      H                ;HL->STRING
                                   TST      C                ;CHECK BASE 10 EXP[
   0035'   79             +.IFN C                -A, [    MOV      A,C                  ]
   0036'   B7             +               ORA      A]
   0037'   CC 00C6'                CZ       FRAC             ;.NNNN
```

```
EF      -
CNVFS - CONVERT FLOATING TO STRING


003A'   FC 00BB'            CM      FRACPT          ;.OONNN
003D'   C4 00B2'            CNZ     INTPT           ;NNN.NNN
0040'   C9                  RET
                    ;++++
                    ;
                    ; MSGN
                    ; TAKES CARE OF THE SIGN OF THE MANTISSA OF A
                    ; FLOATING POINT NUMBER. IF IT IS POSITIVE, NOTHING
                    ; IS DONE. IF IT IS NEGATIVE, A '-' IS PUT IN THE
                    ; STRING AND THE SIGN OF THE FLOATING NUMBER ON
                    ; THE TOP OF THE NCU STACK IS CHANGED
                    ;
                    ; NEEDS:
                    ;   HL -> WHERE TO PUT SIGN IN STRING
                    ;   NCU -> FLOATING POINT NUMBER
                    ;
                    ; RETURNS:
                    ;   NCU -> POSITIVE FLOATING POINT NUMBER
                    ;   HL -> AFTER SIGN (IF ANY)
                    ;
                    ; CALLS:
                    ;   GETMSB - GET EXPONENT
                    ;   DONCU - DO NCU OPERATION
                    ;
                    ;----
0041'   CD 0000:0D  MSGN:   CALL    GETMSB          ;GET EXPONENT
0044'   E680                ANI     N.MSGN          ;CHECK MANTISSA SIGN
0046'   C8                  RZ                      ;POSITIVE? LEAVE IT
0047'   362D                MVI     M,'-'           ;SIGN IN STRING
0049'   23                  INX     H               ;AFTER THE SIGN
004A'   3E15                MVI     A,N.CHSF        ;GET ABSOLUTE VALUE
004C'   C3 0000:09          JMP     DONCU

                    ;++++
                    ;
                    ; NORMLIZ, NORMLZ
                    ; NORMALIZES A FLOATING POINT NUMBER WITH RESPECT TO
                    ; BASE 10
                    ;
                    ; NEEDS:
                    ;   NCU->NUMBER .NNNNNN * 10**M
                    ;   BC = BASE 10 EXPONENT (NORMLZ ONLY)
                    ;
                    ; RETURNS:
                    ;   NCU-> BASE 10 MANTISSA .NNNNNN
                    ;   BC = BASE 10 EXPONENT + M
                    ;
                    ;----
004F'               NORMLIZ:
004F'   01 0000             LXI     B,0             ;BASE 10 EXP = 0
0052'   3E17                MVI     A,N.PTOF        ;IF IS 0,
0054'   CD 0000:09          CALL    DONCU           ;RETURN
                            TST     A[
0057'   B7          +       ORA     A]
0058'   C8                  RZ
```

```
EF      -
CNVFS - CONVERT FLOATING TO STRING


0059'   CD 0000:0D              CALL    GETMSB          ;GET EXPONENT
005C'   FE40                    CPI     N.ESGN          ;EXPONENT NEGATIVE?
005E'   3804                    JRC     ..N1            ;NO? LEAVE IT BE
0060'   FE56                    CPI     #N.MSGN&-42     ;IS TOO SMALL?
0062'   3814                    JRC     NORMLZ          ;DONT ROUND THEN
0064'   C6EB            ..N1:   ADI     -21             ;SHIFT TO THE RIGHT
0066'   E67F                    ANI     #N.MSGN         ;KILL THE SIGN
0068'   57                      MOV     D,A
0069'   1EE0                    MVI     E,0E0H          ;.111 * 2**-21
006B'   D5                      PUSH    D
006C'   CD 0000:11              CALL    IPSH0
006F'   D1                      POP     D
0070'   CD 0000:12              CALL    IPUSH
0073'   3E10                    MVI     A,N.FADD        ;ROUND IT UP
0075'   CD 0000:09              CALL    DONCU
                        ;+
                        ; NORMLZ - ACTUAL NORMALIZATION
                        ; THE BASE 2 EXPONENT IS CHECKED
                        ;   IF IT IS 0 - WE EXIT
                        ;   IF IT IS > 0 - WE DIVIDE BY 10 & CHECK AGAIN
                        ;-
0078'   CD 0000:0D     NORMLZ: CALL    GETMSB          ;GET EXPONENT
                                TST     A               ;IS IT 0?[
007B'   B7              +       ORA     A]
007C'   C8                      RZ                      ;RETURN IF SO
007D'   E640                    ANI     N.ESGN          ;IS IT NEGATIVE?
007F'   2826                    JRZ     ..ND            ;DIVIDE BY 10
                        ;+
                        ; BASE 2 EXPONENT IS < 0. IF THE NUMBER WE HAVE IS
                        ; LARGER THAN OR EQUAL TO .1, WE CAN EXIT. OTHERWISE
                        ; WE MUST MULTIPLY BY 10 AND CHECK AGAIN
                        ;-
0081'   3E17            ..NM:   MVI     A,N.PTOF        ;SAVE NUMBER
0083'   CD 0000:09              CALL    DONCU
0086'   E5                      PUSH    H               ;SUBTRACT .1
0087'   21 035D'                LXI     H,PT1           ;HL->.1
008A'   CD 0000:0C              CALL    FPUSH           ;ON NCU STACK
008D'   E1                      POP     H
008E'   3E11                    MVI     A,N.FSUB        ;SUBTRACT TO
0090'   CD 0000:09              CALL    DONCU           ;COMPARE THEM
                                TST     A               ;SAVE CC'S[
0093'   B7              +       ORA     A]
0094'   F5                      PUSH    PSW
0095'   3E18                    MVI     A,N.POPF        ;KILL RESULT
0097'   CD 0000:09              CALL    DONCU
009A'   F1                      POP     PSW             ;GET CC'S
009B'   F0                      RP                      ;RETURN IF >= .1
009C'   CD 0000:18              CALL    PSHF10          ;ELSE PUSH 10.
009F'   0B                      DCX     B               ;DEC BASE 10 EXP
00A0'   3E12                    MVI     A,N.FMUL        ;AND MULTIPLY
00A2'   CD 0000:09              CALL    DONCU
00A5'   18DA                    JMPR    ..NM            ;CHECK AGAIN
                        ;+
                        ; BASE 2 EXPONENT > 0, WE DIVIDE BY 10 AND
                        ; CHECK AGAIN
```

```
                        ;-
00A7'  CD 0000:18       ..ND:   CALL    PSHF10          ;PUSH 10.
00AA'  3E13                     MVI     A,N.FDIV        ;AND DIVIDE
00AC'  CD 0000:09               CALL    DONCU
00AF'  03                       INX     B               ;BUMP BASE 10 EXP
00B0'  18C6                     JMPR    NORMLZ
                        ;++++
                        ;
                        ; INTPT
                        ; PUTS THE INTEGER PART OF A FLOATING POINT NUMBER
                        ; INTO THE GIVEN STRING
                        ;
                        ; NEEDS:
                        ;   DE -> FIRST DIGIT OF NUMBER IN BUFFER
                        ;   HL -> NEXT BYTE IN STRING TO STORE INTO
                        ;   A = BASE 10 EXPONENT (IF <=0, RETURN)
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER INTEGER PART (IF ANY)
                        ;   DE -> AFTER LAST INTEGER DIGIT
                        ;   B DESTROYED
                        ;
                        ;----
00B2'  47               INTPT:  MOV     B,A             ;B=BASE 10 EXP
00B3'  1A               ..I1:   LDAX    D               ;A=NEXT DIGIT
00B4'  13                       INX     D
00B5'  77                       MOV     M,A             ;COPY DIGIT
00B6'  23                       INX     H
00B7'  10FA                     DJNZ    ..I1            ;UNTIL ALL DONE
00B9'  180B                     JMPR    FRAC            ;DO FSACTION
                        ;++++
                        ;
                        ; FRACPT
                        ; COPIES THE REST OF THE FRACUIONAL DIGITS FROM
                        ; BUFFER INTO STRING. DECIMAL POINT ADDED, TRAILING
                        ; ZEROS SUPPRESSED
                        ;
                        ; NEEDS:
                        ;   DE -> FIRST FRACTIONAL DIGIT
                        ;   HL -> WHERE TO PUT DIGITS IN STRING
                        ;
                        ; RETURNS:
                        ;   DE -> AFTER LAST DIGIT
                        ;   HL -> NEXT BYTE IN STRING
                        ;
                        ;----
00BB'  362E             FRACPT: MVI     M,'.'           ;PUT IN DEC PT
00BD'  23                       INX     H
00BE'  3630             ..F1:   MVI     M,'0'           ;LEADING ZEROS
00C0'  23                       INX     H  .
00C1'  3C                       INR     A               ;BUMP EXPONENT
00C2'  20FA                     JRNZ    ..F1            ;UNTIL 0
00C4'  1803                     JMPR    FRACX           ;REST OF DIGITS
                        ;+
                        ; FRAC - ENTRY POINT WHEN LEADING ZEROS NOT DESIRED
```

```
EF      -
CNVFS - CONVERT FLOATING TO STRING


                        ;-
00C6'   362E            FRAC:   MVI     M,'.'           ;DECIMAL POINT
00C8'   23                      INX     H
00C9'   1A              FRACX:  LDAX    D               ;GET A DIGIT
00CA'   13                      INX     D
00CB'   77                      MOV     M,A             ;COPY IT
00CC'   23                      INX     H
                                TST     A               ;WAS IT NULL?[
00CD'   B7              +       ORA     A]
00CE'   20F9                    JRNZ    FRACX           ;NO? CONTINUE
00D0'   2B                      DCX     H               ;AT NULL
00D1'   3E30                    MVI     A,'0'
00D3'   2B               ..F2:  DCX     H               ;DEC PTR
00D4'   BE                      CMP     M               ;TRAILING 0?
00D5'   28FC                    JRZ     ..F2            ;KILL IT
00D7'   7E                      MOV     A,M             ;A=LAST DIGIT
00D8'   FE2E                    CPI     '.'             ;WAS THE DP?
00DA'   2008                    JRNZ    ..F3            ;NO? EXIT
00DC'   2B                      DCX     H               ;HL->BEFORE DP
                                TST     M               ;IS NULL?[
00DD'   7E              +.IFN M         -A, [   MOV     A,M                     ]
00DE'   B7              +       ORA     A]
00DF'   2003                    JRNZ    ..F3            ;NO? EXIT
00E1'   23                      INX     H               ;POINT AT DP
00E2'   3630                    MVI     M,'0'           ;MAKE IT 0
00E4'   23               ..F3:  INX     H               ;HL->LAST 0 OR NULL
                                CLR     A               ;SET Z BIT[
00E5'   AF              +.IFN A         -A, [   MVI     A                       ,0][
                        XRA     A]]
00E6'   C9                      RET
                        ;++++
                        ;
                        ; FLTOUT
                        ; CONVERT NORMALIZED FLOATING POINT NUMBER TO A
                        ; DIGIT STRING (NO DECIMAL POINT, ASSUMED TO BE
                        ; AT THE FAR LEFT)
                        ;
                        ; NEEDS:
                        ;   HL -> BUFFER TO STORE!AT LEAST 6 DIGITS & NULL
                        ;   NCU-> BASE 10 NORMALIZED FLOATING NUMBER
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER LAST DIGIT
                        ;   DESTROYS B, NCU
                        ;
                        ;----
00E7'   0605            FLTOUT: MVI     B,5             ;FIELD WIDTH
00E9'   E5                      PUSH    H               ;PUSH 10**6
00EA'   21 0361'                LXI     H,TEN6          ;ON NCU STACK
00ED'   CD 0000:0C              CALL    FPUSH
00F0'   E1                      POP     H
00F1'   3E12                    MVI     A,N.FMUL        ;GET 7 DIGITS
00F3'   CD 0000:09              CALL    DONCU
00F6'   3E1E                    MVI     A,N.FIXD        ;CONVERT TO INT
00F8'   CD 0000:09              CALL    DONCU
```

EF -
CNVFS - CONVERT FLOATING TO STRING

```
00FB'  C3 0000:08          JMP     CVFSTR          ;PUT IN STRING
                        ;++++
                        ;
                        ; SCINO
                        ; PUTS A NORMALIZED FLOATING NUMBER INTO SCIENTIFIC
                        ; NOUATION IN A STRING .NNNNNNNENNN
                        ;
                        ; NEEDS:
                        ;   HL -> AFTER SIGN IN STRING
                        ;   NCU-> FLOATING MANTISSA OF NUMBER
                        ;   BC = BASE 10 EXPONENT
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER EXPONENT DIGITS
                        ;   BC, DE, NCU DESTROYED
                        ;
                        ;----
00FE'  C5               SCINO:  PUSH    B               ;SAVE BASE 10 EXP
00FF'  362E                     MVI     M,'.'           ;DECIMAL POINT
0101'  23                       INX     H
0102'  CD 00E7'                 CALL    FLTOUT          ;GET 7 DIGITS
0105'  3645                     MVI     M,'E'           ;EXPONENT
0107'  23                       INX     H
0108'  D1                       POP     D               ;DE = BASE 10 EXP
0109'  CD 0000:12               CALL    IPUSH           ;DOUBLE IT
                                TST     D               ;CHECK HI BYTE[
010C'  7A               +.IFN D         -A, [   MOV     A,D             ]
010D'  B7               +               ORA     A]
010E'  11 0000                  LXI     D,0             ;ASSUME POS
0111'  2803                     JRZ     ..S1            ;OK
0113'  11 FFFF                  LXI     D,-1            ;NEGATIVE
0116'  CD 0000:12       ..S1:   CALL    IPUSH           ;PUSH HI BYTE
0119'  C3 0000:07               JMP     CVDSTR          ;INTO STRING
                        ;++++
                        ;
                        ; GETNUM
                        ; PARSES A FLOATING POINT NUMBER FROM A STRING
                        ;
                        ; NEEDS:
                        ;   HL -> FIRST DIGIT OF STRING
                        ;
                        ; RETURNS:
                        ;   FAIL:
                        ;       C BIT SET
                        ;       FAILS IF NO DIGITS AT ALL
                        ;   SUCCEED:
                        ;       HL -> AFTER LAST DIGIT, EXPONENT, ETC.
                        ;       TOP OF NCU STACK HAS GLOATING NUMBER
                        ;       A = $FVAL (TYPE FLOAT)
                        ;   DESTROYS DE, BC
                        ;
                        ; CALLS:
                        ;   CARTYP - CHECK CHARACTER TYPE
                        ;   DONCU - DO!NCU OPERATION
                        ;   PSHF10!- PUSH FLOATING 10
```

EF -
CONVERT STRING TO FLOATING

```
                        ;   GETINT, GETI - GET INUEGER FROM STRING
                        ;   GETEXP - PARSE EXPONEOT
                        ;
                        ;----
011C'   7E              GETNUM: MOV     A,M             ;CHECK FOR DECIMAL
011D'   FE2E                    CPI     '.'             ;POINT FIRST
011F'   2805                    JRZ     ..G1            ;OK TO START
0121'   CD 0000:06             CALL    CARTYP          ;CHECK 1ST CHAR
0124'   3065                    JRNC    FAIL            ;FAIL IF NO DIGIT
                        ;+
                        ; PARSE INTEGER PART. IF A DECIMAL POINT IS FOUND,
                        ; AN EXPONENT IS COMPUTED IN B. THIS EXPONENT IS
                        ; DECREMENTED!EVERY TIME A DIGIT AFTER THE DECIMAL
                        ; POINT IS FOUND
                        ;-
0126'   CD 0159'        ..G1:   CALL    GETINT          ;PARSE INTEGER PART
0129'   0600                    MVI     B,0             ;EXPONENT IS 0
012B'   7E                      MOV     A,M             ;CHECK FOR DECIMAL
012C'   FE2E                    CPI     '.'             ;POINT
012E'   2004                    JRNZ    ..G2            ;ISNT ONE, TOO BAD
0130'   23                      INX     H               ;HL->AFTER OP
0131'   CD 0176'                CALL    GETI            ;PARSE FRACTION
                        ;+
                        ; IF AN EXPONENT IS FOUND, IT IS PARSED AND
                        ; ADDED TO THE CURRENT EXPONENT IN B
                        ;-
0134'   CD 018D'        ..G2:   CALL    GETEXP          ;PARSE EXPONENT
0137'   3E1C                    MVI     A,N.FLTD        ;CONVERT TO FLT
0139'   CD 0000:09             CALL    DONCU
                        ;+
                        ; HERE B = BASE 10 EXPONENT AND THE STACK HAS
                        ; THE FLOATING NUMBER PARSED WITH THE DECIMAL
                        ; POINT TO THE FAR RIGHT. WE MULTIPLY/DIVIDE
                        ; THE NUMBER OF TIMES INDICATED BY THE EXPONENT
                        ; TO GET THE TRUE VALUE
                        ;-
013C'                   ..G3:   TST     B               ;EXPONENT 0?[
013C'   78              +.IFN B         -A, [   MOV     A,B             ]
013D'   B7              +               ORA     A]
013E'   2816                    JRZ     ..EX            ;YES? CAN EXIT
0140'   FA 014E'               JM      ..G5            ;NEGATIVE? DIVIDE
0143'   CD 0000:18             CALL    PSHF10          ;MULTIPLY BY 10.0
0146'   3E12                    MVI     A,N.FMUL
0148'   05                      DCR     B               ;DEC EXPONENT
0149'   CD 0000:09      ..G4:   CALL    DONCU           ;DO IT
014C'   18EE                    JMPR    ..G3            ;GO AGAIN
014E'   CD 0000:18      ..G5:   CALL    PSHF10          ;DIVIDE BY 10.0
0151'   3E13                    MVI     A,N.FDIV
0153'   04                      INR     B               ;BUMP EXPONENT
0154'   18F3                    JMPR    ..G4
0156'   3E06              ..EX:   MVI     A,$FVAL         ;RETURN TYPE FLT
0158'   C9                      RET
                        ;++++
                        ;
                        ; GETINT, GETI
```

```
                        ; INTERNAL ROUTINES TO PARSE INTEGER AND MAINTAIN
                        ; A 32 BIT NUMBER ON NCU STACK. THE EXPONENT IN
                        ; B IS DECREMENTED EVERY TIME A DIGIT IS PARSED
                        ;
                        ; NEEDS:
                        ;   HL -> DIGIT
                        ;   NCU STACK HAS CURRENT NUMBER FOR GETI, GETINT
                        ;   WILL PUSH A 32 BIT 0
                        ;
                        ; RETURNS:
                        ;   FAIL:
                        ;       C BIT SET
                        ;       FAILS IF NO DIGIT FOUND
                        ;   SUCCEED:
                        ;       HL -> NON-DIGIT
                        ;       NCU STACK HAS 32 BIT INTEGER PARSED
                        ;
                        ; CALL:
                        ;   CARTYP - CHECHK CHARACTER TYPE
                        ;   DONCU - DO NCU OPERATION
                        ;   IPSHO - PUSH INTEGER 0
                        ;   IPUSH - PUSH INTEGER
                        ;
                        ;----
0159'  CD 0000:11       GETINT: CALL    IPSHO           ;PUSH 32 BIT ZERO
015C'  CD 0000:11               CALL    IPSHO
015F'  CD 0000:06               CALL    CARTYP          ;CHECK DIGIT
0162'  3027                     JRNC    FAIL            ;FAIL RETURN
0164'  7E               GI:     MOV     A,M             ;A=DIGIT
0165'  23                       INX     H               ;HL->AFTER DIGIT
0166'  D630                     SUI     '0'             ;GET IN BINARY
                                CLR     D               ;PUT IT IN DE[
0168'  1600             +.IFN D         -A, [   MVI     D                ,0][
                         XRA    A]]
016A'  5F                       MOV     E,A
016B'  CD 0000:12               CALL    IPUSH           ;PUSH ON NCU STACK
016E'  CD 0000:11               CALL    IPSHO           ;PAD HI ORDER
0171'  3E2C                     MVI     A,N.DADD        ;ADD TO RESULT
0173'  CD 0000:09               CALL    DONCU
0176'  CD 0000:06       GETI:   CALL    CARTYP          ;CHECK NEXT DIGIT
0179'  D0                       RNC                     ;RETURN IF NOT DIG
017A'  11 000A                  LXI     D,10            ;PUSH INTEGER 10
017D'  CD 0000:12               CALL    IPUSH           ;ON NCU STACK
0180'  CD 0000:11               CALL    IPSHO
0183'  3E2E                     MVI     A,N.DMUL        ;MULTIPLY OUR NUM
0185'  CD 0000:09               CALL    DONCU           ;BY 10
0188'  05                       DCR     B               ;DEC EXPONENT
0189'  18D9                     JMPR    GI              ;CONTINUE
018B'  37               FAIL:   STC                     ;SET CARRY
018C'  C9                       RET
                        ;++++
                        ;
                        ; GETEXP
                        ; PARSE EXPONENT IN STRING AND RETURN IN B
                        ;
```

```
EF      -
CONVERT STRING TO FLOATING


                        ; NEEDS:
                        ;   HL -> POSSIBLE EXPONENT IN STRING
                        ;  B = OLD EXPONENT OR 0
                        ;
                        ; RETURNS:
                        ;   HL -> AFTER EXPONENT (IF ANY)
                        ;   B = OLD EXPONENT PLUS NEW EXPONENT
                        ;
                        ; CALLS:
                        ;   GETINT - PARSE INTEGER
                        ;   DONCU - DO NCU OPERATION
                        ;   IPOP - POP INTEGER
                        ;
                        ;----
018D'   FE45            GETEXP: CPI     'E'             ;IS EXPONENT?
018F'   C0                      RNZ                     ;NO? SKIP IT
0190'   C5                      PUSH    B               ;SAVE OLD EXP
0191'   23                      INX     H               ;HL->AFTER E
0192'   7E                      MOV     A,M             ;A = NEXT THING AFTER E
0193'   FE2D                    CPI     '-'             ;CHECK SIGN
0195'   F5                      PUSH    PSW             ;SAVE CCS
0196'   2001                    JRNZ    ..G1            ;NO SIGN?
0198'   23                      INX     H               ;HL->AFTER SIGN
0199'   CD 0159'        ..G1:   CALL    GETINT          ;GET EXPONENT
019C'   CD 0000:10             CALL    IPOP            ;POP HI PART
019F'   CD 0000:10             CALL    IPOP            ;DE=EXPONENT
01A2'   F1                      POP     PSW             ;WAS THERE A SIGN?
01A3'   7B                      MOV     A,E             ;A=EXPONENT PARSED
01A4'   2002                    JRNZ    ..G2            ;NO SIGN
01A6'   ED44                    NEG                     ;NEGATE IT
01A8'   C1              ..G2:   POP     B               ;B=OLD EXPONENT
01A9'   80                      ADD     B               ;ADD TO NEW ONE
01AA'   47                      MOV     B,A             ;SAVE IN B
01AB'   C9                      RET                     ;AND RETURN
                        ;++++
                        ;
                        ; POWER
                        ;
                        ;----
                                M.CMD[POWER,..POWX,0,PLAY,..PARG,..POW]C
01AC'   18              +POWER: .BYTE $CMDADR
01AD'   03              +       .BYTE (..POWX-POWER)/16+1
01AE'   00              +       .BYTE 0
01AF'   0000:16         +       .WORD PLAY
01B1'   01BC'           +       .WORD ..POW
01B3'   01BF'           +       .WORD ..PARG
01B5'   504F57455200+           .ASCIZ /POWER/
                        +]
01BB'   00                      .BYTE   0
01BC'   CD 0000:05      ..POW:  CALL    ARG
01BF'   060600          ..PARG: .BYTE   $FVAL,$FVAL,$TAF
01C2'   CD 0000:13              CALL    IYNCU           ;GET 1ST ARG
01C5'   CD 0000:0F              CALL    INC5IY          ;POINT AT 2ND ONE
01C8'   CD 0000:13              CALL    IYNCU           ;AND 2ND ARG
01CB'   3E0B                    MVI     A,N.PWR         ;POWER FUNCTION
```

```
01CD'   181B                    JMPR    FRET            ;AND RETURN
01CF'                   ..POWX:
                        ;++++
                        ;
                        ; SIN
                        ;
                        ;----
                                M.CMD[SINE,SINX,0,SQRT,FNARG,..SIN1][
01CF'   18              +SINE:  .BYTE $CMDADR
01D0'   03              +       .BYTE (SINX-SINE)/16+1
01D1'   00              +       .BYTE 0
01D2'   0277'           +       .WORD SQRT
01D4'   01DE'           +       .WORD ..SIN1
01D6'   01E4'           +       .WORD FNARG
01D8'   53494E4500      +       .ASCIZ /SINE/
                        +]
01DD'   00                      .BYTE   0
01DE'                   ..SIN1:
01DE'   3E02                    MVI     A,N.SIN         ;SINE
01E0'   F5              FFUN:   PUSH    PSW             ;SAVE OPERATION
01E1'   CD 0000:05              CALL    ARG             ;GET ARGUMENT
01E4'   0600            FNARG:  .BYTE   $FVAL,$TAF
01E6'   CD 0000:13              CALL    IYNCU           ;ONTO NCU FROM IY
01E9'   F1                      POP     PSW             ;A = FUNCTION
01EA'   CD 0000:09      FRET:   CALL    DONCU           ;RESULT ON NCU
01ED'   3E06                    MVI     A,$FVAL         ;FLOATING VALUE
01EF'   C3 0000:1B              JMP     RETNONE         ;RETURN
01F2'                   SINX:
                        ;++++
                        ;
                        ; COS
                        ;
                        ;----
                                M.CMD[COSINE,..COSX,0,AFIX[C],FNARG,..COS1][
01F2'   18              +COSINE: .BYTE $CMDADR
01F3'   02              +       .BYTE (..COSX-COSINE)/16+1
01F4'   00              +       .BYTE 0
01F5'   630C            +       .WORD AFIX[C][("C"-101Q)*16+FSTINT]
01F7'   0203'           +       .WORD ..COS1
01F9'   01E4'           +       .WORD FNARG
01FB'   434F53494E45+           .ASCIZ /COSINE/
                        +]
0202'   00                      .BYTE   0
0203'   3E03            ..COS1: MVI     A,N.COS ;COSINE
0205'   18D9                    JMPR    FFUN
0207'                   ..COSX:
                        ;++++
                        ;
                        ; TAN
                        ;
                        ;----
                                M.CMD[TANGENT,..TANX,0,AFIX[T],FNARG,..TAN][
0207'   18              +TANGENT:       .BYTE $CMDADR
0208'   02              +       .BYTE (..TANX-TANGENT)/16+1
0209'   00              +       .BYTE 0
```

```
020A'   641C          +         .WORD AFIX[T][("T"-101Q)*16+FSTINT]
020C'   0219'         +         .WORD ..TAN
020E'   01E4'         +         .WORD FNARG
0210'   54414E47454E+           .ASCIZ /TANGENT/
                      +]
0218'   00                      .BYTE   0
0219'   3E04          ..TAN:    MVI     A,N.TAN          ;TANGENT
021B'   18C3                    JMPR    FFUN
021D'                 ..TANX:
'                     ;++++
                      ;
                      ;  INT
                      ;
                      ;----
                                M.CMD[INT,..INTX,0,AFIX[I],..INTA,..INT][
021D'   18            +INT:     .BYTE $CMDADR
021E'   02            +         .BYTE (..INTX-INT)/16+1
021F'   00            +         .BYTE 0
0220'   636C          +         .WORD AFIX[I][("I"-101Q)*16+FSTINT]
0222'   022B'          +         .WORD ..INT
0224'   022E'          +         .WORD ..INTA
0226'   494E5400      +         .ASCIZ /INT/
                      +]
022A'   00                      .BYTE   0
022B'   CD 0000:05    ..INT:    CALL    ARG
022E'   0400          ..INTA:   .BYTE   $IVAL,$TAF
0230'   CD 0000:0E             CALL    GETOPND          ;GET OPERAND
0233'   C3 0000:1B             JMP     RETNONE          ;RETURN INTEGER
0236'                 ..INTX:
                      ;++++
                      ;
                      ;  ASIN
                      ;
                      ;----
                                M.CMD[ARCSIN,..ACSX,0,AFIX[A],FNARG,..ASIN][
0236'   18            +ARCSIN:  .BYTE $CMDADR
0237'   02            +         .BYTE (..ACSX-ARCSIN)/16+1
0238'   00            +         .BYTE 0
0239'   62EC          +         .WORD AFIX[A][("A"-101Q)*16+FSTINT]
023B'   0247'         +         .WORD ..ASIN
023D'   01E4'         +         .WORD FNARG
023F'   41524353494E+           .ASCIZ /ARCSIN/
                      +]
0246'   00                      .BYTE   0
0247'   3E05          ..ASIN:   MVI     A,N.ASIN         ;ARC SINE
0249'   1895                    JMPR    FFUN
024B'                 ..ACSX:
                      ;++++
                      ;
                      ;  ACOS
                      ;
                      ;----
                                M.CMD[ARCCOS,..ACCX,0,ARCSIN,FNARG,..ACOS][
024B'   18            +ARCCOS:  .BYTE $CMDADR
024C'   02            +         .BYTE (..ACCX-ARCCOS)/16+1
```

```
024D'   00              +           .BYTE 0
024E'   0236'           +           .WORD ARCSIN
0250'   025C'           +           .WORD ..ACOS
0252'   01E4'           +           .WORD FNARG
0254'   415243434F53+    +           .ASCIZ /ARCCOS/
                        +]
025B'   00                          .BYTE   0
025C'   3E06             ..ACOS:   MVI     A,N.ACOS          ;ARC COSINE
025E'   C3 01E0'                    JMP     FFUN
0261'                     ..ACCX:
                          ;++++
                          ;
                          ; ATAN
                          ;
                          ;----
                                    M.CMD[ARCTAN,..ATNX,0,ARCCOS,FNARG,..ATAN][
0261'   18              +ARCTAN:  .BYTE $CMDADR
0262'   02              +         .BYTE (..ATNX-ARCTAN)/16+1
0263'   00              +         .BYTE 0
0264'   024B'           +         .WORD ARCCOS
0266'   0272'           +         .WORD ..ATAN
0268'   01E4'           +         .WORD FNARG
026A'   41524354414E+    +         .ASCIZ /ARCTAN/
                        +]
0271'   00                          .BYTE   0
0272'   3E07             ..ATAN:   MVI     A,N.ATAN          ;ARC TANGENT
0274'   C3 01E0'                    JMP     FFUN
0277'                     ..ATNX:
                          ;++++
                          ;
                          ; SQRT
                          ;
                          ;----
                                    M.CMD[SQRT,..SQRX,0,SUBSTR,FNARG,..SQRT][
0277'   18              +SQRT:    .BYTE $CMDADR
0278'   02              +         .BYTE (..SQRX-SQRT)/16+1
0279'   00              +         .BYTE 0
027A'   0000:1C         +         .WORD SUBSTR
027C'   0286'           +         .WORD ..SQRT
027E'   01E4'           +         .WORD FNARG
0280'   5351525400      +         .ASCIZ /SQRT/
                        +]
0285'   00                          .BYTE   0
0286'   3E01             ..SQRT:   MVI     A,N.SQRT          ;SQUARE ROOT
0288'   C3 01E0'                    JMP     FFUN
028B'                     ..SQRX:
                          ;++++
                          ;
                          ; LOG
                          ;
                          ;----
                                    M.CMD[LOG,..LOGX,0,LPAD,FNARG,..LOG][
028B'   18              +LOG:     .BYTE $CMDADR
028C'   02              +         .BYTE (..LOGX-LOG)/16+1
028D'   00              +         .BYTE 0
```

```
028E'   0000:14     +          .WORD LPAD
0290'   0299'       +          .WORD ..LOG
0292'   01E4'       +          .WORD FNARG
0294'   4C4F4700    +          .ASCIZ /LOG/
                    +]
0298'   00                     .BYTE   0
0299'   3E08         ..LOG:   MVI     A,N.LOG          ;BASE 10 LOG
029B'   C3 01E0'               JMP     FFUN
029E'                 ..LOGX:
                      ;++++
                      ;
                      ; LN
                      ;
                      ;----
                               M.CMD[LN,..LNX,0,LOG,FNARG,..LN][
029E'   18          +LN:       .BYTE $CMDADR
029F'   02          +          .BYTE (..LNX-LN)/16+1
02A0'   00          +          .BYTE 0
02A1'   028B'       +          .WORD LOG
02A3'   02AB'       +          .WORD ..LN
02A5'   01E4'       +          .WORD FNARG
02A7'   4C4E00      +          .ASCIZ /LN/
                    +]
02AA'   00                     .BYTE   0
02AB'   3E09          ..LN:    MVI     A,N.LN           ;BASE E LOG
02AD'   C3 01E0'               JMP     FFUN
02B0'                 ..LNX:
                      ;++++
                      ;
                      ; EXP
                      ;
                      ;----
                               M.CMD[EXP,..EXPX,0,AFIX[E],FNARG,..EXP][
02B0'   18          +EXP:      .BYTE $CMDADR
02B1'   02          +          .BYTE (..EXPX-EXP)/16+1
02B2'   00          +          .BYTE 0
02B3'   632C        +          .WORD AFIX[E][("E"-101Q)*16+FSTINT]
02B5'   02BE'       +          .WORD ..EXP
02B7'   01E4'       +          .WORD FNARG
02B9'   45585000    +          .ASCIZ /EXP/
                    +]
02BD'   00                     .BYTE   0
02BE'   3E0A          ..EXP:   MVI     A,N.EXP          ;RAISE TO E POWER
02C0'   C3 01E0'               JMP     FFUN
02C3'                 ..EXPX:
                      ;++++
                      ;
                      ; PI
                      ;
                      ;----
                               M.CMD[PI,..PIX,0,AFIX[P],0,..PI][
02C3'   18          +PI:       .BYTE $CMDADR
02C4'   02          +          .BYTE (..PIX-PI)/16+1
02C5'   00          +          .BYTE 0
02C6'   63DC        +          .WORD AFIX[P][("P"-101Q)*16+FSTINT]
```

```
02C8'   02D0'       +         .WORD ..PI
02CA'   0000        +         .WORD 0
02CC'   504900      +         .ASCIZ /PI/
                    +]
02CF'   00                    .BYTE   0
02D0'   3E1A         ..PI:    MVI     A,N.PUPI          ;RETURN PI
02D2'   C3 01EA'              JMP     FRET
02D5'                ..PIX:
                     ;++++
                     ;
                     ; FORMAT
                     ; FORMATS A FLOATING POINT NUMBER INTO AN INTEGER AND FR
                     ACTIONAL
                     ; PART (ASCII STRING) WITH N DIGITS AFTER THE DECIMAL PO
                     INT
                     ;
                     ; NEEDS:
                     ;   FLOATING NUMBER TO FORMAT
                     ;   INTEGER GIVING NUMBER OF DIGITS AFTER DECIMAL POINT
                     ;       (MAY BE BETWEEN 0 AND 6)
                     ;
                     ; RETURNS:
                     ;   STRING WITH FORMATTED NUMBER
                     ;
                     ; CALLS:
                     ;   ARG - FETCH ARGUMENTS
                     ;   ALSTS - ALLOCATE STRING
                     ;   GETOPND - FETCH OPERAND
                     ;   INC5IY - BUMP IY TO NEXT ARG
                     ;   MSGN - GET SIGN
                     ;   NORMLIZ - NORMALIZ NUMBER
                     ;   NUMIT - CONVERT TO ASCII
                     ;
                     ;----
                              M.CMD[FORMAT,..FMTX,0,AFIX[F],..FMARG,..FMT][
02D5'   18           +FORMAT: .BYTE $CMDADR
02D6'   09           +        .BYTE (..FMTX-FORMAT)/16+1
02D7'   00           +        .BYTE 0
02D8'   633C         +        .WORD AFIX[F][("F"-101Q)*16+FSTINT]
02DA'   02E5'        +        .WORD ..FMT
02DC'   02E8'        +        .WORD ..FMARG
02DE'   464F524D4154+         .ASCIZ /FORMAT/
                     +]
02E5'   CD 0000:05    ..FMT:  CALL    ARG
02E8'   060400        ..FMARG:.BYTE   $FVAL,$IVAL,0
02EB'   E5                    PUSH    H
02EC'   3E01                  MVI     A,($SASCII+19)/16
02EE'   CD 0000:04            CALL    ALSTS             ;ALLOCATE STRING
02F1'   E5                    PUSH    H                 ;SAVE THE PTR
02F2'   EB                    XCHG                      ;TO THE TOP
02F3'   CD 0000:0E            CALL    GETOPND           ;GET FLOATING VALUE
02F6'   CD 0000:0F            CALL    INC5IY            ;POINT TO NEXT ARG
                     ;+
                     ; GET # OF DIGITS AFTER DP IN DE
                     ; IT MUST BE POSITIVE AND LESS THAN 6
```

```
                        ;-
02F9'  CD 0000:0E               CALL    GETOPND         ;GET # DIGITS AFTER DP
02FC'  7B                       MOV     A,E             ;A = # DIGITS AFTER
                                TST     A               ;MUST BE > 0[
02FD'  B7               +       ORA     A]
02FE'  FA 0305'                 JM      ..ERR           ;ERROR
0301'  FE07                     CPI     6+1             ;MUST BE < 6
0303'  3804                     JRC     ..NN            ;OR ERROR
0305'                   ..ERR:  ERROR   ER.DP[
0305'  CD 0000:0B       +       CALL ERRPGM
0308'  37               +       .BYTE ER.DP
                        +]
                        ;+
                        ; PRINT INITIAL SIGN AND NORMALIZE NUMBER
                        ;-
0309'  CD 0041'         ..NN:   CALL    MSGN            ;GET THE SIGN
030C'  D5                       PUSH    D               ;SAVE DIGIT COUNT
030D'  CD 004F'                 CALL    NORMLIZ         ;NORMALIZE NUMBER
                        ;+
                        ; IF THE NUMBER IS GREATER THAN 10**10, IT IS
                        ; TOO BIG TO DISPLAY. IF IT IS GREATER THAN 10**6,
                        ; TRAILING ZEROS ARE ADDED
                        ;-
0310'  79                       MOV     A,C             ;A=EXPONENT
0311'  FE07                     CPI     6+1             ;0 < EXP < 6?
0313'  381C                     JRC     ..N1            ;YES?
0315'  FE0B                     CPI     10+1            ;6 < EXP < 10?
0317'  3808                     JRC     ..N             ;YES?
0319'  FEF0                     CPI     -16             ;-16 < EXP < 10
031B'  3014                     JRNC    ..N1            ;YES?
                                ERROR   ER.FMT          ;ELSE ERROR[
031D'  CD 0000:0B       +       CALL ERRPGM
0320'  38               +       .BYTE ER.FMT
                        +]
0321'  DE05             ..N:    SBI     5               ;SUBTRACT 5
0323'  4F                       MOV     C,A             ;NEW EXPONENT
0324'  CD 00E7'                 CALL    FLTOUT
0327'  3630             ..NW:   MVI     M,'0'           ;TRAILING 0S
0329'  23                       INX     H
032A'  0D                       DCR     C               ;UNTIL COUNTUP
032B'  20FA                     JRNZ    ..NW
032D'  362E                     MVI     M,'.'           ;PUT DP IN
032F'  180C                     JMPR    ..N5
                        ;+
                        ; CONVERT FLOATING NUMBER TO ASCII AND GET THE
                        ; NUMBER OF DIGITS REQUESTED AFTER THE DECIMAL POINT
                        ;-
0331'  E5               ..N1:   PUSH    H               ;SAVE PTR
0332'  CD 002B'                 CALL    NUMIT           ;GET ASCII NUMBER
0335'  E1                       POP     H               ;HL -> FIRST DIGIT
                        ;+
                        ; WE SEARCH THE DIGIT STRING FOR THE DECIMAL POINT
                        ;-
0336'  3E2E                     MVI     A,'.'           ;DP TO COMPARE
0338'  BE               ..N2:   CMP     M               ;IS IT POINT?
```

```
0339'   23                      INX     H               ;POINT AFTER
033A'   20FC                    JRNZ    ..N2            ;SCAN FOR IT
033C'   2B                      DCX     H               ;HL -> AT DP
                        ;+
                        ; DECIMAL POINT FOUND, WE COUNT OUT THE DIGITS
                        ; UNTIL DECIMAL POINT COUNTER EXHAUSTED OR END
                        ; OF STRING IS REACHED
                        ; E = NUMBER OF DIGITS ALLOWED AFTER DP
                        ;-
033D'   D1              ..N5:   POP     D
                                TST     E               ;IF 0, EXIT[
033E'   7B              +.IFN E         -A, [   MOV     A,E             ]
033F'   B7              +               ORA     A]
0340'   2812                    JRZ     ..OUT
0342'   23                      INX     H               ;HL -> AFTER DP
0343'   1D              ..N3:   DCR     E               ;ONE LESS DIGIT
0344'   FA 0354'                JM      ..OUT           ;ALL GONE? EXIT
                                TST     M               ;FOUND THE NULL?[
0347'   7E              +.IFN M         -A, [   MOV     A,M             ]
0348'   B7              +               ORA     A]
0349'   23                      INX     H               ;HL -> AT NEXT CHAR
034A'   20F7                    JRNZ    ..N3            ;NO? CONTINUE
                        ;+
                        ; END OF STRING REACHED, NUMBER MUST BE PADDED WITH
                        ; TRAILING ZEROS UNTIL THE DECIMAL POINT COUNTER
                        ; IN E BECOMES MINUS
                        ;-
034C'   2B                      DCX     H               ;HL -> AT NULL
034D'   3630            ..N4:   MVI     M,'0'           ;TRAILING 0
034F'   23                      INX     H
0350'   1D                      DCR     E               ;ONE LESS DIGIT
0351'   F2 034D'                JP      ..N4
                        ;+
                        ; EXIT POINT
                        ; PUT IN TRAILING NULL, POP OFF STRING ADDRS & EXIT
                        ;-
0354'   D1              ..OUT:  POP     D               ;DE -> TOP OF STRING
                                CLR     M               ;ENDING NULL[
0355'   3600            +.IFN M         -A, [   MVI     M               ,0][
                        XRA     A]]
0357'   E1                      POP     H               ;RESTORE OTHERS
0358'   3E08                    MVI     A,$STRADR       ;RETURN TYPE STRING
035A'   C3 0000:1B             JMP     RETNONE         ;RETURN
035D'                   ..FMTX:
035D'   CCCC    7DCC      1:    .WORD   0CCCCH,07DCCH   ;.1
0361'   2400    14F4      N6:   .WORD   02400H,014F4H   ;10**6
                        .END
```

EF -

+++++ SYMBOL TABLE +++++

```
AASN    005F          ALSTS   0000:04 X     ARCCOS  024B'   I     ARCSIN  0236'   I
ARCTAN  0261'   I     ARG     0000:05 X     ARGSTK  60CC          BACKGR  65CC
BLANK   0020          BOTRAM  6000          BOTTOM  64A9          CARTYP  0000:06 X
CHARSL  65DD          CLEARS  60CA          CNTRL   000C          CNTRLC  65CD
CNTRLO  65D9          CNTRLU  0015          CNTRLZ  65E4          CNVFS   0000'   I
COSINE  01F2'   I     CPLARE  611C          CPLSIZ  0140          CR      000D
CSBLOK  668D          CSFLAG  6260          CURCX   6814          CURCY   6816
CURREN  64A5          CVDSTR  0000:07 X     CVFSTR  0000:08 X     C.C0    0011
C.C1    0012          C.CX    000B          C.CY    000D          C.DP    0013
C.ST    0002          C.X     0003          C.XF    000F          C.XS    0007
C.Y     0005          C.YF    0010          C.YS    0009          DDTON   65CE
DEVBL   6589          DEVCL0  6579          DEVCL1  657B          DEVCL2  657D
DEVCL3  657F          DEVCL4  6581          DEVCL5  6583          DEVCL6  6585
DEVCL7  6587          DEVFB   65CB          DEVHCB  658F          DEVMO   658B
DEVNM   65B7          DEVNT   658D          DEVTNA  65BB          DEVTNB  65BF
DEVTNC  65C3          DEVVA   65BD          DEVVAR  6579          DEVVB   65C1
DEVVBL  6591          DEVVC   65C5          DEVVD   65C9          DEVVN   65B9
DEVVS   65C7          DEVXCD  65B3          DEVYCD  65B5          DOLPLH  62E8
DOLPPT  62EA          DONCU   0000:09 X     DUMBST  6577          EDBCNT  64AD
EDCCNT  64A7          EDLONG  6812          EDMODE  681C          EDNAME  64A1
EDNCX   680C          EDNCY   680E          EDNEWS  64A5          EDOCX   6808
EDOCY   680A          EDPN    6806          EDPO    6804          EDPTRC  64AB
EDPTRL  64A9          EDSTR   681A          ENDSTR  0000:0A X     ERABIT  0002
ERRPGM  0000:0B X     ER.ARA  002F          ER.ARG  0034          ER.ASN  0015
ER.BOX  001A          ER.CHN  0002          ER.CMD  001F          ER.CNV  0016
ER.COR  001B          ER.CTL  0036          ER.DEL  0026          ER.DIM  0030
ER.DIV  0018          ER.DP   0037          ER.DSK  0019          ER.EDT  0035
ER.FMT  0038          ER.FNF  001C          ER.FOR  0028          ER.IMP  0003
ER.LAB  0025          ER.MAC  0022          ER.NAE  002D          ER.NAM  0029
ER.NEG  003A          ER.NOT  0023          ER.NUL  0039          ER.NUM  002B
ER.NXT  001D          ER.OFL  0017          ER.OPN  0014          ER.OVE  001E
ER.PAR  002A          ER.REN  002C          ER.RET  0024          ER.SEP  0021
ER.SNP  0031          ER.SPC  002E          ER.STK  0004          ER.SW   0032
ER.TER  0020          ER.UFL  0033          ER.UNF  0027          EXP     02B0'   I
EXTDEL  002E          E.HVAL  0002          E.LVAL  0001          E.SIZ   0005
E.TYP   0000          E.VAL   0001          FAIL    018B'         FCNTH   65D2
FCNTI   65D3          FCNTJ   65D4          FCNTK   65D5          FCNTL   65D6
FCNTV   65E0          FCNTY   65E3          FFUN    01E0'         FIRST   64A3
FLAGS   65CB          FLTOUT  00E7'         FNARG   01E4'         FOREGR  65D0
FORMAT  02D5'   I     FPUSH   0000:0C X     FRAC    00C6'         FRACPT  00BB'
FRACX   00C9'         FRAGSI  0400          FREELS  65E5          FRET    01EA'
FSTDOL  648C          FSTINT  62EC          FWDPTR  65E7          GETEXP  018D'
GETI    0176'         GETINT  0159'         GETMSB  0000:0D X     GETNUM  011C'   I
GETOPN  0000:0E X     GI      0164'         HCAREA  6593          INC5IY  0000:0F X
INCRO   65E9          INT     021D'   I     INTPT   00B2'         IPOP    0000:10 X
IPSHO   0000:11 X     IPUSH   0000:12 X     IYNCU   0000:13 X     JUNK    6542
KBLOCK  65F1          KEYFLG  67FF          KEYPTK  6533          KEYTRK  67F7
LF      000A          LISTON  65E2          LN      029E'   I     LOG     028B'   I
LPAD    0000:14 X     MACSTU  6536          MACTOP  625E          MAXFRG  0040
MNMX    65EB          MSGN    0041'         M.POPS  0078          NBLKB   0000
NBLKM   0001          NCUCOM  0029          NCUDAT  0028          NCUIY   0000:15 X
NEWBOT  6810          NL      000A          NORMLI  004F'         NORMLZ  0078'
NSADDR  6802          NUMBUF  64A3          NUMIT   002B'         N.ACOS  0006
N.ARG   0003          N.ASIN  0005          N.ATAN  0007          N.CHSD  0034
N.CHSF  0015          N.CHSS  0074          N.COS   0003          N.DADD  002C
```

EF -

+++++ SYMBOL TABLE +++++

```
N.DDIV 002F        N.DIV  0004        N.DMUL 002E        N.DMUU 0036
N.DSUB 002D        N.EMAX 0017        N.EMIN FF69        N.ESGN 0040
N.EXP  000A        N.FADD 0010        N.FDIV 0013        N.FIXD 001E
N.FIXS 001F        N.FLTD 001C        N.FLTS 001D        N.FMUL 0012
N.FSUB 0011        N.LN   0009        N.LOG  0008        N.MSGN 0080
N.NOP  0000        N.OFLO 0001        N.POPD 0038        N.POPF 0018
N.PTOD 0037        N.PTOF 0017        N.PTOS 0077        N.PUPI 001A
N.PWR  000B        N.SADD 006C        N.SDIV 006F        N.SIGN 0040
N.SIN  0002        N.SMUL 006E        N.SMUU 0076        N.SQRT 0001
N.SSUB 006D        N.TAN  0004        N.UFLO 0002        N.XCHD 0039
N.XCHF 0019        N.XCHS 0079        N.ZERO 0020        OLDCHR 64A0
OLDCUR 649F        OLDKEY 649D        OLDXY  65EF        ONEBUF 6729
OPRL   0014        OPRSP  655B        OPRSTK 6547        OPRSZ  655D
OUTOFF 62E7        O.CHAR 0000        O.PREC 0002        O.SIZ  0006
O.SUB  0004        O.TYP  0003        PCNT   655E        PI     02C3'   I
PIXVAL 65ED        PLAY   0000:16 X   POINTE 64A7        PONOFF 65DA
POPOPN 0000:17 X   POWER  01AC'   I   PRINTR 6540        PSHF10 0000:18 X
PSHOPN 0000:19 X   PT1    035D'       PUTOPN 0000:1A X   RAMEND 7FFF
RAMSTR 6900        RANSHT 655F        RETNON 0000:1B X   RMDTMP 6538
RUBACK 0005        RUBOUT 007F        SAVESP 625C        SCINO  00FE'
SCNEED 0004        SCRWIN 67CD        SINE   01CF'   I   SINX   01F2'
SOPRSP 653D        SOPRSZ 653F        SQRT   0277'   I   STACK  60C8
STAKTO 6000        STRSIZ 6800        SUBSTR 0000:1C X   SUBSTU 6534
TAB    0009        TANGEN 0207'   I   TAPBUF 64B3        TAPCON 64AF
TAPPRO 64B1        TBFEND 6533        TEMPHD 60CA        TEMPS  62E5
TEN6   0361'       TMPARG 67AD        TOP    6818        TTYBEG 6261
TTYEND 62E1        TTYINT 62E3        TTYPTR 62E1        TXTWIN 67E2
UARTFL 649C        USREND 65F1        V3PTR  6573        VDCHAR 649E
VDNLF  65CF        VIPLH  6545        VOICE0 6563        WRMODE 65EE
ZGIM2  0001        ZGREND 681D        $AADR  0010        $ADDRF 0007
$ADDRI 0005        $ADDRS 0009        $ANY   001A        $ANYNA FFFC
$ANYVA FFFE        $ARGPT 0011        $BGPTR 000F        $BNDL  0007
$CALLE 000D        $CMDAD 0018        $CPLBL 002A        $CSBLO 0028
$DATAP 0007        $DOLDE 0001        $DOLOO 0002        $DVAL  0000
$END   001C        $FADR  000E        $FLAGS 0002        $FORBL 0024
$FORPT 000B        $FVAL  0006        $GOSUB 001A        $IADR  000C
$INPBU 0018        $INPPT 0016        $IVAL  0004        $KEYBL 0026
$LENGT 0001        $LINPT 0005        $LOCPT 0009        $MIBBL 0022
$MIBEN 001B        $NAMAD 000A        $NAME  000A        $NASCI 0009
$NDEL  0080        $NLINK 0003        $NULL  0002        $NVALU 0005
$REPEA 001E        $RVSTU 0013        $SAME  0020        $SASCI 000A
$SLEN  0006        $STRAD 0008        $STRPT 0003        $TAF   0000
$TYPE  0000        $USE   0005        .BLNK. 0000:03 X   .DATA. 0000*    X
.PROG. 0365'   X
```

3 ERRORS WERE DETECTED *****

```
.REMARK /
                        *
                       * * *
                        ***
                      *******
                         *
                        *
                       *
                      *
                     *
                    ********

      "WHEN YOU CARE ENOUGH TO PROGRAM
                THE VERY BEST"

               ZGRASS V2.00000000
BY JAY FENTON, NOLA DONATO, AND TOM DEFANTI
                    (C) 1978

/
.XLIST
;THIS MODULE CONTAINS ALL THE SYSTEM EQUATES
;EQUATES PREFACED BY $ SIGNS ARE GENERALLY OFFSETS

;MACRO INVOCATION BLOCK (MIB) EQUATES:

$TYPE=0             ;TYPE BYTE
$LENGTH=1           ;LENGTH BYTE (FILLED BY ALLOC)
```