GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station

PROJECT INITIATION

Date: July 10, 1970

Project Title: To develop Specifications for FOCAL-11

Project No.: A-1264

Project Director: John C. Alderman

Sponsor: Digital Equipment Corporation - PDP-11

Effective June 1, 1970 Estimated to run until: August 31, 1970

Type Agreement: Industrial Research Amount: $ 2,500

Reports: ~~Monthly Progress~~
Final

Contact Person: Mr. Andrew C. Knowles
Product Line Manager
PDP-11 Product Line
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Assigned to Nuclear & Biological Sciences Division

COPIES TO:

- ☐ Project Director
- ☐ Director
- ☐ Associate Director
- ☐ Assistant Director(s)
- ☐ Division Chiefs
- ☐ Branch Head
- ☐ General Office Services
- ☐ Engineering Design Services
- ☐ Photographic Laboratory
- ☐ Research Security Officer
- ☐ Accounting
- ☐ Purchasing
- ☐ Report Section
- ☑ Library
- ☐ Rich Electronic Computer Center
- ☐ _____

4(12-66)

GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station

PROJECT TERMINATION

Date 3/31/71

PROJECT TITLE: To Develop Specifications for FOCAL-11

PROJECT NO: A-1264

PROJECT DIRECTOR: Mr. J. C. Alderman

SPONSOR: Digital Equipment Corporation

TERMINATION EFFECTIVE: 3/31/71

CHARGES SHOULD CLEAR ACCOUNTING BY: All charges cleared.

Nuclear & Biological Sciences Division

COPIES TO:

Project Director
Director
Associate Director
Assistant Directors
Division Chief
Branch Head
Accounting
Engineering Design Services
General Office Services
Photographic Laboratory
Purchasing
Report Section
Library
Security
Rich Electronic Computer Center

ES 402 (R10-62)

Progress Report on the Development of FOCAL-11

PROJECT NAME: FOCAL-11 SPECIFICATIONS

PROJECT NUMBER: A1264

AUTHORS: GWEN McALLEN
JOHN ALDERMAN

Progress Report on the Development of FOCAL-11

I. SPECIFICATIONS

The preliminary specifications for FOCAL-11 have been prepared according to the format set forth in Programming Projects Control Standard, the programming department memo #005-003-004-000. These specifications include the following sections:

1.0 OVERALL DESCRIPTION
2.0 HARDWARE ENVIRONMENT
3.0 SOFTWARE ENVIRONMENT
7.0 LANGUAGE

These sections constitute a basic framework for the language.

II. CODING

Roughly, one-third of the coding has been completed and is ready to be tested. The commands that have been coded are: ON, IF, DO, GOTO, FOR, SET, ASK, TYPE.

In order for these commands to be executed, the following subroutines are necessary: READC, PRINTC, PUSHF, POPF, SORTC, SORTJ, PROCESS, TESTC, TESTN, MOVTORØ, GETC, SPNOR, GETNSC, FINDLN, GETARG, SORTV, TSTLPR, TSTGRP, GETLN, ECALL, EVAL, GTOPR, MOVSØØ, GETOP. All of these subroutines have been coded.

The subroutines EVAL, MOVSØØ, GETOPR, and GETOP in the present version of FOCAL-11 are similar to the subroutines of the same names in BASIC-11. The other subroutines are similar to corresponding subroutines in FOCAL-8 (1969). In fact, considerable effort was made so that FOCAL-11 would be consistent with FOCAL-8, a feature that is beneficial to both the experienced and inexperienced users of the FOCAL language. The following conventions have been adopted in order to implement consistency.

Macros Being Utilized	Meaning of the Macros in PAL-11	Comparable FOCAL Macro
PUSH DUMMY1	= MOV DUMMY1, -(SP)	PUSHA
POP DUMMY1	= MOV (SP)+, DUMMY1	POPA
PUSHJ DUMMY1	= JSR R5, DUMMY1	PUSHJ
POPJ	= RTS R5	POPJ
GETC	= MOV (RA)+, CHAR	GETC
SORTC	= JSR R5, XSORTC	SORTC
SORTJ	= JSR R5, XSORTB	SORTJ
PRINTC	= JSR R5, XPRINTC	PRINTC
GETLN	= JSR R5, XGETLN	GETLN
FINDLN	= JSR R5, XFIND	FINDLN
TESTC	= JSR R5, XTESTC	TESTC
TESTN	= JSR R5, XTESTN	TESTN
PUSHF	= JSR R5, XPUSHF	PUSHF
POPF	= JSR R5, XPOPF	POPF
TSTGRP	= JSR R5, TSTGRP	TSTGRP
TSTLPR	= JSR R5, LPRTST	TSTLPR
SPNOR	= JSR R5, XNSC	SPNOR

SP = the stack pointer (register 6) and is the register that handles the push-down list.
R5 = register 5 and is used in most of the subroutine calls.

The present version of FOCAL-11 has been designed so that the PAL-11A Assembler will read in and store text. Thus the actual input form of a FOCAL-11 program must conform to the conventions of PAL-11A. In addition only the teletype can be used for input and output. Also, indirect statements are the only type of statements that are allowed in the present coding. For example, suppose one wants to run the following FOCAL-11 program.

```
1.1Ø  TYPE 2,9
2.13  SET X = 5
2.15  QUIT
```

In order to use the PAL-11A to store this program, one would type the following statements on the teletype keyboard to the assembler.

```
        TXTBGN = #L1.Ø1
L1.Ø1:  .WORD  L2.13
        .BYTE  1,Ø1
        .ASCII 'TYPE 2,9'
        .EVEN
        .BYTE  15, 12          ; CR, LF
L2.13:  .WORD  L2.15
        .BYTE   2, 13
        .ASCII 'SET X = 5'
        .EVEN
        .BYTE  15, 12          ; CR LF
L2.15:  .WORD  Ø                ; A zero signifies the end of text.
        .BYTE  2, 15
        .ASCII 'QUIT'
        .EVEN
        .BYTE  15, 12
```

The syntactical form required to start execution of this program remains to be defined.

FOCAL-11

THE FORMULA CALCULATOR

FOR THE D.E.C. PDP-11

AUTHORS: GWEN MCALLEN
JOHN ALDERMAN

PROJECT NUMBER: A1264

```
;VARIABLES IN FOCAL-11

        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        R6=%6
        R7=%7
        CHAR=R3                 ;TEXT CHARACTER PLACED IN R3.
        SORTCN=R2               ;IN SORTC - SORTCN=  X (0 BYTES
                                ;DOWN IN LIST MATCH OCCURRED).
                                ;IN TESTN - IF CHAR IS A NUMBER,
                                ;SORTCN = ITS BINARY EQUIVALENT.
ISLN:   0                       ;ADDRESS OF LINE FOUND IN FINDLN.
STLN:   0                       ;ADDRESS OF PREVIOUS LINE TESTED
                                ;IN FINDLN.
CDE:    0                       ;OPERATOR CODE IN IF,ON COMMANDS.
                                ;OCTAL VALUES ARE: "<"=1,"="=2,
                                ;">"=4.
C:      0                       ;PROGRAM COUNTER FOR TEXT.
XTP:XPC: 0                      ;TEXTPOINTERS - XPC POINTS TO THE
                                ;  1ST WORD OF A LINE IN TEXT.
I:                              ;ADDITIONAL TEXTPOINTER.
EM:                             ;       "       "
        XRT=R2                  ;EXTRA REGISTER.
GSW:    1                       ;FLAG FOR DO COMMAND: GROUP=0,
                                ;LINE=100000, ALL=1.
NENO    0                       ;LINE NUMBER READ IN DO COMMAND.
I:      0                       ;VARIABLE POINTER.
SW:     0                       ;ASK - TYPE SWITCH.
BGSW:   1                       ;DEBUG SWITCH - NONZERO FOR
                                ;LITERAL.
R:      15                      ;LIST BRANCHER.
SW:     0                       ;POINTER FOR OUTPUT FORMAT.
TONE:   1                       ;FLOATING-POINT (NORMALIZED)
                                ;1.0 (3 WORDS).

AC:     0                       ;LOWEST ADDRESS OF 3 CONSECUTIVE WORDS.

ARG:    0                       ;SAME.

AC1:    0                       ;SAME.

AC2:    0                       ;SAME.

LX:     144                     ;FLTING-PT. (NORMALIZED) 144
                                ;  (3 WORDS).

D1:     0                       ;LOCATION TO STORE VARIABLE NAME.
BS:     0                       ;LOCATION TO STORE VARIABLE SUBSCRIPT.
OP:     0                       ;FUNCTION CODE.
SUB:    0                       ;FLAG: =0 (GETC), <>0 (READC).
```

;MNEMONICS IN FOCAL-11

```
;PUSH DUMMY1        MOV DUMMY1,-(R6)
;POP DUMMY1         MOV (R6)+,DUMMY1
;PUSHJ DUMMY1       JSR R5,DUMMY1
;POPJ               RTS R5
;GETC               MOVB (R4)+,CHAR
;PACKC              MOV CHAR,ADD1
;SORTC              JSR R5,XSORTC
;SORTJ              JSR R5,SORTB
;PRINTC             JSR R5,XPRNTC
;GETLN              JSR R5,XGETLN
;FINDLN             JSR R5,XFIND
;GETNSC             JSR R5,XNSC
;TESTC              JSR R5,XTESTC
;TESTN              JSR R5,XTESTN
;BUMP RN            TST (RN)+
;PUSHF              JSR R5,XPUSHF
;POPF               JSR R5,XPOPF
;TSTGRP             JSRR5,TSTGRP
;SKP                BR .+2
;SKP2               BR .+4
;SPNOR              JSR R5,XSPNOR
;TSTLPR             JSR R5,LPRTST
;SORTV              JSR R5,XSORTV
;MOVSTX             JSR PC,MOVS00
;MOVTOR0            JSR PC,MOVTOR0
; READC             JSR R5, XREADC
; GTOPR             JSR PC, GTPR00
; GETOP             JSR PC, GTP00
```

```
        RESET=2
        INIT=1
        WAITR=4
        WRITE=12
        READ=11
        RSLOT=0
        PSLOT=1
V:      1
V:      2
1:      1                       ;SIZE OF BUFFER1 (IN BYTES).
        2                       ;FORMAT (ANSCII UNFORMATTED).
        1                       ;BYTE-COUNT.
        .=.+1                   ;RESERVE 1 BYTE OF STORAGE.
        .EVEN
2:      1                       ;SIZE OF BUFFER2 (IN BYTES).
        2                       ;FORMAT (ANSCII UNFORMATTED).
        1                       ;BYTE-COUNT.
        .=.+1                   ;RESERVE 1 BYTE OF STORAGE.
        .EVEN
RT:     IOT                     ;RESET DEVICES TO INITIAL
         .WORD 0                ;STATE.
         .BYTE RESET,0
        IOT                     ;ASSOCIATE DEVICES WITH SLOT
         .WORD RDEV             ;NUMBERS (RDEV,RSLOT).
         .BYTE INIT,RSLOT
        IOT                     ;(PDEV,PSLOT).
         .WORD PDEV
         .BYTE INIT,PSLOT
        MOV #TXTBGN,CFRS        ;SETUP PERMANENT  POINTER TO
        MOV CFRS,XPC            ;BEGINNING OF TEXT AND A CHANGE-
        MOV XPC,R4              ;ABLE LINE PNTR. AND CHARAC. PNTR.
ONT:    JSR R5,PROCA            ;EXECUTE PROGRAM.
        MOV XPC,R4              ;ADJUST POINTERS TO NEXT LINE.
        MOV (R4),XPC
        BR XCONT

;TEXT IS STORED SUCH THAT THE FIRST WORD OF A LINE IS
;A POINTER TO THE NEXT LINE AND THE SECOND WORD IN
;THE LINE CONTAINS THE LINE NUMBER. THE REST OF THE LINE
;CONTAINS TEXT.


;READC - XREADC USES R3.
;XREADC READS IN A CHARACTER FROM THE KEYBOARD AND
;PLACES IT IN R3 (CHAR).

EADC:   IOT                     ;IS TELETYPE KEYBOARD READY?
         .WORD XREADC
         .BYTE WAITR,RSLOT      ;NO - GOTO XREADC.
        IOT                     ;YES - READ IN CHARACTER
         .WORD BUF1             ;FROM KEYBOARD AND PLACE
         .BYTE READ,RSLOT       ;IN BUFFER1.
        MOVB BUF1+6,CHAR        ;MOVE CHARCTER TO CHAR.
        RTS R5
```

```
        ;PRINTC - XPRNTC, PRINTC ON THE TELETYPE PRINTER
        ;THE CHARACTER IN CHAR (R3).

NTC:    PUSH R0
        PUSH R1
        PUSH R2
P:      IOT                     ;IS TELETYPE PRINTER READY?
         .WORD LOOP
         .BYTE WAITR,PSLOT      ;NO - GOTO LOOP.
        SORTJ                   ;YES - SORT CHARACTER AGAINST
         .WORD TLISTB           ;TERMINATOR LIST (CR,↑K).
        .WORD SPRNT
        PUSHJ XOUTC             ;NO MATCH - PRINT THE CHARACTER.
        POP R2
        POP R1
        POP R0
        POPJ

TC:     MOVB CHAR,BUF2+6        ;MOVE CHAR INTO BUFFER2 AND
        IOT                     ;PRINT
         .WORD BUF2             ;THE CHARACTER.
         .BYTE WRITE,PSLOT
        POPJ

        ;XOUTL - PRINTS ONLY THE CHARACTER IN CHAR WITHOUT
        ;SORTING THE CHARACTER AGAINST A LIST FIRST.

TL:     IOT
        .WORD XOUTL
         .BYTE WAITR,PSLOT
        PUSHJ XOUTC
        POPJ

        ;CRPRNT - CARRIAGE RETURN,LINE FEED PRINT.

PRNT:   IOT
         .WORD BUFCR
         .WORD WRITE,PSLOT
        POPJ

        ;CKPRNT - CONTROL/K CAUSES THE FOLLOWING CHARACTERS
        ;TO BE PRINTED: ↑, K, CR, LF, AND 6 SPACES.

PRNT:   IOT
         .WORD BUFCK
         .WORD WRITE, PSLOT
        POPJ

FCR:    2                       ;SIZE OF BUFFER IN BYTES.
        2                       ;FORMAT - ANSCII UNFORMATTED.
        2                       ;BYTE-COUNT.
        .BYTE 15,12             ;CR,LF.

F CK:   12                      ;SIZE OF BUFFER IN BYTES.
        2                       ;FORMAT - ANSCII UNFORMATTED.
        12                      ;BYTE-COUNT.
         .BYTE 136,113,15,12    ;↑, K, CR, LF.
         .BYTE 40,40,40,40,40,40        ;6 SPACES.
```

```
;PUSHF - XPUSHF, PUSHES A FLOATING-PT. NUMBER (WHOSE
;ADDRESS IS IN REGISTER R5) ONTO THE STACK. XPUSHF
;USES REGISTERS R0,R5. THE 3RD WORD OF THE FLOATING-
;PT. NUMBER IS ON THE TOP OF THE STACK.

SHF: MOV R0,XR0          ;SAVE DATA IN R0.
     MOV (R5)+,R0
     PUSH (R0)+          ;PUSH THE 1ST WORD FIRST.
     PUSH (R0)+          ;PUSH THE 2ND WORD NEXT.
     PUSH (R0)           ;PUSH THE 3RD WORD LAST.
     MOV XR0,R0          ;RESTORE DATA TO R0.
     RTS R5

;POPF - XPOPF, POPS A FLOATING-PT. NUMBER FROM THE
;STACK TO THE LOCATION GIVEN IN REGISTER 5. XPOPF
;USES REGISTERS R0,R5. NOTE THAT THE 3RD WORD OF THE
;FLOATING-PT. NUMBER IS ON THE TOP OF THE STACK. THEREFORE
;THE 3RD WORD COMES OFF THE STACK FIRST.

?P: MOV R0,XR0           ;SAVE DATA IN R0.
    MOV (R5)+,R0
    ADD #6,R0            ;GET ADDRESS OF 3RD WORD + 2.
    POP -(R0)            ;POP 3RD WORD TO 3RD WORD ADDRESS.
    POP -(R0)            ;POP 2ND WORD NEXT.
    POP -(R0)            ;POP 1ST WORD LAST.
    MOV XR0,R0           ;RESTORE DATA TO R0.
    RTS R5
```

```
;SORTC - XSORTC, UPON ENTRY: A CHARACTER IS IN R3 (CHAR)
;AND THE POINTER TO A LIST ADDRESS IS IN R5. XSORTC COM-
;PARES THE CHARACTER IN R3 TO THOSE IN A LIST. IF A MATCH
;OCCURS, THEN R2 (SORTCN) IS SET = 2 TIMES THE NUMBER OF
;BYTES DOWN THE LIST THAT THE MATCH OCCURRED. IF A MATCH
;OCCURS IN SORTC, THE 1ST RETURN IS TAKEN. IF NO
;MATCH OCCURS, THEN THE 2ND RETURN IS TAKEN.
;THE CALLING SEQUENCE FOR XSORTC IS
;SORTC
;  .WORD LISTADDRESS
;XXX                     ;1ST RETURN - A MATCH OCCURRED.
;YYY                     ;2ND RETURN - NO MATCH.
;SORTC USES REGISTERS R0,R1, R2,R5. SORTCN=R2, CHAR=R3.

RTC: MOV (R5)+,R2        ;MOVE LIST ADDRESS TO R2.
     MOV R2,R0
     INC R0
EST: MOVB (R2)+,R1       ;MOVE CHARACTER IN LIST TO R1.
     BMI SEXC            ;IF CHARAC IS < 0, GOTO SEXC.
     CMPB CHAR,R1        ;COMPARE CHAR WITH R1.
     BNE SRTEST          ;NOT EQUAL - GO TO SRTEST.
     SUB R0,R2           ;EQUAL - SET UP SORTCN.
     SORTCN=R2
     ADD SORTCN,SORTCN
     RTS R5

;SORTJ - SORTB. UPON ENTRY: A CHARACTER IS IN R3 (CHAR),
;THE POINTER TO A LIST IS IN R5, AND A POINTER TO A 2ND
;LIST IS IN (R5)+2. SORTB COMPARES THE CHARACTER IN R3
H OCCURRED, THOSE IN THE 1ST LIST. IF A
;LOCATION, SORTCN + ADDRESS OF LIST2, CONTAINS AN ADDRESS
;WHICH IS BRANCHED TO. IF NO MATCH OCCURS, THEN A RETURN
;IS EXECUTED TO THE 3RD LINE AFTERTHE CALL TO SORTB.
;THE CALLING SEQUENCE FOR SORTJ IS
;SORTJ
;  .WORD LIST1ADDRESS
;  .WORD LIST2ADDRESS
H OCCURRED              ;RETURN HERE IF NO
;SORTJ CALLS SORTC AND USES REGISTER R5 PLUS THOSE USED
;IN SORTC.

TB:  MOV (R5)+,R2
     PUSHJ XSORTC+2     ;SORT CHAR AGAINST LIST.
     BR SRTAD           ;RETURN HERE IF MATCH FOUND.
C:   BUMP R5            ;NO MATCH - RETURN TO 3RD
     RTS R5             ;LINE AFTER CALL.
AD:  MOV (R5),R5        ;ADDRESS OF COMMAND SET UP.
     ADD SORTCN,R5
     MOV (R5),R5        ;GO DO COMMAND.
     RTS R5
```

```
;GOTO COMMAND. UPON ENTRY R4 POINTS TO A LINE NUMBER.
;GOTO PLACES THIS NUMBERIN THE LOCATION "LINENO" AND PUTS
;THE ADDRESS OF THIS LINE IN XPC.
;GOTO CALLS GETLN AND FINDLN.

C:  GETLN
    FINDLN
    ERROR                   ;LINE NOT FOUND.
    MOV THISLN,XPC          ;PUT ADDRESS OF LINE IN P.C.
    MOV XPC,R4

;PROCA - UPON ENTRY XPC AND R4 POINT TO THE FIRST
;WORD OF A LINE IN THE STORED TEXT. UPON ENTRY TO PROCESS
;R4, THE CHARACTER POINTER, NOW POINTS TO THE 3RD WORD
;OF THE LINE . PROCA CONSIDERS ONLY THE 1ST LETTER OF A
;COMMAND AND PROCEEDS TO THE ADDRESS OF THE APPROPRIATE
;COMMAND (IF THE COMMAND IS LEGAL). UPON EXIT R3 CONTAINS
;THE COMMAND TERMINATOR AND R4 POINTS TO THE NEXT CHARACTER.
;PROCA CALLS GETC AND SORTC AND USES REGISTERS R3,R4,
;R5 PLUS THOSE IN SORTC.

CA: ADD #4,R4               ;SETUP CHARACTER POINTER.
CESS: GETC                  ;GET CHARACTER.
C:  CMP #15,CHAR            ;IS CHAR A C.R.?
    BNE .+4         ;->
:   POPJ            ; I     ;YES - EXIT "PROCESS".
    SORTC           ;<-     ;NO.
     .WORD GLIST
    BR PROCESS              ;IGNORE SPACE, ",", ↑K AND ";".
    BIC #40,CHAR            ;CONVERT LOWER CASE LETTER TO
    PUSH CHAR               ;UPPER CASE.
1:  GETC
    SORTC
     .WORD GLIST
    BR .+4          ;->     ;SKIP NEXT LINE IF C.T. FOUND.
    BR PRC1         ; I     ;IGNORE CHARACTERS UP TO C.T.
    PUSH CHAR       ;<-
    MOV 2(R6),CHAR          ;GET 1ST LETTER OF COMMAND.
    SORTC                   ;GET ADDRESS OF COMMAND.
     .WORD COMLIST          ;ADDRESS OF COMMAND LIST.
    BR .+4          ;->
    ERROR           ; I     ;ILLEGAL COMMAND.
    POP CHAR        ;<-
    MOV #COMGO,(R6)
    ADD SORTCN,(R6)
    JMP @(R6)+              ;GO DO COMMAND.
```

```
;ON,IF COMMAND. UPON ENTRY: R4 POINTS TO THE
;CHARACTER FOLLOWING THE COMMAND TERMINATOR, SORTCN
;(R2) IS SUCH THAT THE LOCATION, SORTCN + #COMGO, CONTAINS
;THE ADDRESS OF EITHER THE IF OR ON COMMAND.
;THE OPCDE REGISTER (R0) WILL
;CONTAIN INFORMATION ABOUT THE OPERATOR(S) BETWEEN
;THE FIRST AND SECOND EXPRESSION IN THE 2-BRANCH IF OR
;ON COMMANDS: "<" = 1, "=" = 2, ">" = 4. R2 IS USED AS
;A FLAG. REGISTERS USED - ALL. ON,IF CALL TESTC,GETC,
;GETNSC,SORTC,SORTJ,ECALL.

F:  MOV #COMGO,COMADD      ;DISTINGUISH IF AND ON.
    ADD SORTCN,COMADD
    OPCDE=R0
    CLR OPCDE              ;CLEAR OPERATOR CODE.
    CLR R2
    PUSH R2                ;SAVE FLAG.
    PUSH OPCDE
    TESTC                  ;TEST CHARACTER.
T:  JSR PC,EVAL#0-2        ;T- EVALUATES EXPRESSION.
    BVS OPTX               ;N- OVERFLOW TEST FOR PAREN MATCH.
    BR BOPT                ;F.
T1: GETNSC                 ;A- GET A NONSPACE CHARACTER.
    SORTC                  ;SORT CHAR AGAINST OPERATOR
     .WORD OPLIST          ;LIST.
    SEN
    BGE OPT2               ;MATCH - GOTO OPT2.
    TST 2(R6)              ;NO MATCH;TEST BRANCHING FLAG.
    BEQ IF3BR              ;IF FLAG=0, GOTO IF3BR.
    BR SETBR               ;OTHERWISE DO A 2-BRANCH IF,ON.
T:  MOV #FLAC,R0           ;MOVE NUMBER EVALUATED BY
    MOVTOR0                ;EVAL TO FLAC.
    BR AOPT1
2:  MOVF=EMT+7             ;MOVE NUMBER TO FLARG.
    MOV #FLAC,R1
    MOV #FLARG,R0
    MOVF+FAM
1:  GETNSC                 ;GET NON-SPACE CHARACTER.
    SORTJ                  ;SORT CHAR AGAINST PAREN
     .WORD PROPLS          ;AND OPERATOR LIST.
     .WORD APROP           ;RETURN TO OPT1 IF OPRTR AND
X:  ERROR                  ;TO AOPT IF L-PAREN.(R2<>0)
BR: CMPF=EMT+10
    MOV #FLAC,R1
    MOV #FLARG,R0
    CMPF+FAM               ;RELATIONSHIP BETWEEN EXPR1
    BLT ROT1               ;AND EXPR2 COMPARED WITH OPCDE.
    BEQ ROT2
    ROR OPCDE
2:  ROR OPCDE
1:  ROR OPCDE
    BCS IF2BR              ;GOTO IF2BR IF OPCDE AGREES.
    MOV #-1,R0             ;NO AGREEMENT - MOVE TO TERMI-
    BR IF3                 ;NATOR AND DO NEXT COMMAND.
```

```
BR:  MOV @-2,R0          ;3-BRANCH IF.
     TST FLAG+2          ;TEST VALUE OF EXPR1.
     BGT BR3             ;IF > 0, GO PAST 2 COMMAS.
     BEQ BR2             ;IF = 0, GO PAST 1 COMMA.
     INC R0              ;EXPR1<0 - DO NOT GO PAST
     INC R0              ;ANY COMMAS.
     TST R0
     BMI IF3

BR:  GETLN               ;READ THE LINE NUMBER.
BR1: GETC                ;MOVE TO TERMINATOR.
     SORTC
      .WORD TLISTA
     BR .+4              ;->
     BR IF2BR1           ; I
     CMP COMADD,@IFADD   ;<-

     BNE DO+2            ;@IFADD=POINTER TO IF COMMAND.
     BR GOTO+2           ;WAS THE COMMAND AN IF OR AN ON?
                         ;ON COMMAND = DO.
                         ;IF COMMAND = GOTO.
:    PUSH R0
     SORTJ               ;<-
      .WORD TLIST        ; I     ;LOOK FOR COMMA OR C.T.
      .WORD ILIST        ; I
     GETC                ; I     ;NO MATCH.
     BR .-8              ;->
:    GETC                        ;FOUND A COMMA.
     POP R0
     BR BR2

     DEC 2(SP)                   ;( OR [ ENCOUNTERED. SET FLAG.
     JMP AOPT

     BIS @1,(R6)                 ;< ENCOUNTERED. SET OPCDE.
     BR OPTI

     BIS @2,(R6)                 ;= ENCOUNTERED. SET OPCDE.
     BR OPTI

     BIS @4,(R6)                 ;> ENCOUNTERED. SET OPCDE.
     BR OPTI
```

```
        ;DO USES REGISTERS R2 AND R0.
        PUSH R4                 ;SAVE TEXTPOINTERS.

        GETLN                   ;GET THE LINE NUMBER.
        ~~GETLN~~
        PUSHF
         .WORD TEXTP
RP:     PUSHF                   ;PUSH NAGSW.
         .WORD NAGSW
        TST NAGSW               ;IS NAGSW NEGATIVE?
        BMI DOONE               ;YES - GO TO DOONE.
        FINDLN                  ;NO - FIND LINE WHOSE #
        BR .+2          ;-->    ;IS = TO OR > LINENO.
        MOV THISLN,R2   ;<--
        BUMP R2
        MOV R2,R1               ;GET ADDRESS OF THE LINE #.
        TSTGRP                  ;IS LINE # IN THE GROUP?
        ERROR                   ;NO.
RP1:    PUSHJ PROCA             ;YES - EXECUTE LINE.
        POPF                    ;RESTORE NAGSW
         .WORD NAGSW
        MOV OXPC,R2             ;CHECK FOR END OF TEXT.
        BEQ DCONT               ;YES - GO TO DCONT.
        MOV R2,XPC              ;NO.
        BUMP R2
        TST NAGSW               ;TEST NAGSW FOR DO ALL OR GROUP.
        BGT .+8         ;->     ;DO ALL - SKIP NEXT 3 LINES.
        MOV R2,R1       ; I     ;DO GROUP.
        TSTGRP          ; I     ;IS NEXT LINE IN GROUP?
        BR DCONT        ; I     ;NO - GOTO DCONT.
        MOV (R2),LINENO ;<-     ;YES.
        BR DGRP1
OONE:   FINDLN                  ;FIND THE LINE.
        ERROR                   ;LINE IS NOT FOUND.
        PUSHJ PROCA             ;DO THE LINE.
        POPF                    ;RESTORE THE DATA.
         .WORD NAGSW
CONT:   POPF
         .WORD TEXTP
        POP R4
        JMP PROC
```

```
        ;XTESTC - TESTC - TELLS WHETHER A NON-SPACE CHARACTER (IN
        ;R3) IS A TERMINATOR (1ST RETURN), A NUMBER (2ND RETURN),
        ;A FUNCTION (3RD RETURN), OR AN ALPHABETIC (4TH RETURN).
        ;TESTC CALLS SPNOR,SORTC,TESTN AND USES REGISTERS R3 AND
        ;R5 PLUS THOSE IN THE CALLED SUBROUTINES. UPON ENTRY A
        ;CHARACTER IS IN R3 (CHAR). IF IT IS A SPACE, THEN THE NEXT
        ;NON-SPACE CHARACTER FROM TEXT IS PLACED IN R3 AND
        ;IS TESTED.

ESTC:   SPNOR                   ;IGNORE SPACES.
        SORTC                   ;IS CHAR A TERMINATOR?
          .WORD TERMS
        RTS R5                  ;YES - 1ST RETURN.
        BUMP R5                 ;NO.
        CMP #'F,CHAR            ;IS CHAR AN F?
        BEQ X03                 ;YES - GOTO 3RD RETURN.
        TESTN                   ;NO.
        RTS R5                  ;CHAR IS A "." - 1ST RETURN.
        BR .+4          ;->
        RTS R5          ; I     ;CHAR IS A 0 - 2ND RETURN.
        BUMP R5         ;<-     ;CHAR IS ALPHABETIC - 4TH
3:      BUMP R5                 ;RETURN.
        RTS R5


        ;FOR, SET COMMAND. UPON ENTRY R4 POINTS TO THE CHARACTER
        ;FOLLOWING THE COMMAND TERMINATOR. FOR,SET USES ALL
        ;OF THE REGISTERS AND CALLS GETARG,SPNOR,EVAL,SORTJ,
        ;PUSHF,POPF,PROCESS.

OR:SET:PUSHJ GETARG
        SPNOR                   ;IGNORE SPACES.
        CMP CHAR,#'=            ;IS CHAR AN "="?
        BEQ .+4         ;->
        ERROR           ; I     ;NO.
        PUSH PT1        ;<-      ;YES-SAVE POINTER TO VARIABLE.
        JSR PC,EVAL00           ;GET INITIAL VALUE OF EXPR.
        POP PT1
        MOV PT1,R0              ;PLACE THIS VALUE IN LOCATION
        MOVTOR0                 ;GIVEN BY PT1.
        GETC
        SORTJ                   ;SORT CHARAC AGAINST TERMI-
          .WORD TLIST           ;NATOR LIST.
          .WORD FLIST1
        ERROR
INCR:   PUSH PT1                ;SAVE VARIABLE ADDRESS.
        JSR PC,EVAL00           ;EVALUATE INCREMENT, IF ANY.
        MOV @FLAC,R0
        MOVTOR0
        GETC
        SORTJ                   ;SORT CHARAC AGAINST TERMI-
          .WORD TLIST           ;NATOR LIST.
          .WORD FLIST2
        ERROR
```

```
MIT:  MOV #FLAC,R1
      MOV #FLARG,R0
      MOVF+FAM
      JSR PC,EVALGO              ;EVALUATE LIMIT - NO ERROR
                                 ;DETECTION AFTER LIMIT.

      MOV #FLAC,R0
      MOVTOR0
NT:   PUSHF                      ;SAVE TEXT OF OBJECT STMNTS.
       .WORD TEXTP
      PUSH R4
      PUSHJ PROCESS              ;DO THE OBJECT STMNTS.
      POP R4
      POPF                       ;RESTORE THE REMAINING TEXT.
       .WORD TEXTP
      ADDF=EMT+1                 ;ADD THE INCREMENT TO THE
      MOV #FLARG,R1              ;VARIABLE.
      MOV 2(R6),R0
      ADDF+FAM
      MOV #FLAC,R1               ;TEST LIMIT AGAINST
      CMPF+FAM                   ;VARIABLE.
      BGE FCONT                  ;LIMIT > OR =VAR. GOTO FCONT.
      BUMP R6                    ;LIMIT < VAR.
      POPJ

TOR0: MOV R1,(R0)+
      MOV R2,(R0)+
      MOV R3,(R0)
      JSR PC

FIN:  MOV #FLTONE,R1             ;SET INCREMENT = 1.
      MOV #FLARG,R0              ;FLTONE=NORMALIZED, FLTING
      MOVF+FAM                   ;POINT 1.
      BR FCONT
```

```
                ;ASK, TYPE COMMAND. UPON ENTRY R4 POINTS TO THE CHARACTER
                ;AFTER THE COMMAND TERMINATOR. ENTER TYPE WITH R0 <
                ;CR > 0. ASK, TYPE USES REGISTERS R0, R3, R5.
                ;ASK, TYPE CALLS SORTF, GETARG, PRINTC, FLINPT, EVAL, GETC, SORTJ,
                ;GETLN, XOUTL.

:       CLR R0                  ;DISTINGUISH ASK AND TYPE
E:      MOV R0,ATSW             ;COMMANDS.
K:      CLR DEBGSW              ;RE-ENABLE THE TRACE.
        SPNOR                   ;IGNORE SPACES.
        SORTJ                   ;SPECIAL CHARACTER?
         .WORD ALIST
         .WORD ATLIST
        TST ATSW                ;TEST QUOTE SWITCH.
        BNE TYPE2
        PUSHJ GETARG            ;DO ASK - SETUP PTI.
        PUSH CHAR               ;SAVE IN-LINE CHARACTER.
        MOV #':,R3
        PRINTC                  ;TYPE COLON.
        INC INSUB               ;INDICATE 'READC'.
        MOV #1,R0
        JSR R5,FLINPT           ;READ DATA AND SAVE.
        POP CHAR                ;RETEST LAST TERMINATOR.
        BR ASK
PE2:    JSR PC,EVAL00           ;DO TYPE.
        JSR R5,FOUTPUT          ;PRINT.
        BR TYPE

UOT:    INC DEBGSW              ;DISABLE TRACE.
        BEQ .+4         ;->
        GETC            ; I     ;TYPE LITERALS.
        SORTJ           ;<-
         .WORD TLIST2
         .WORD TLIST3
        PRINTC
        BR TQUOT+2

NTR:    GETC                    ;PASS PERCENT SIGN.
        GETLN                   ;READ FORMAT CONTROL:"Z7.03".
        MOV LINENO,FISW         ;FISW=OUTPUT INFORMATION.
        BR TASK

RLF2:   MOV CCR,R3              ;SPLAT = C.R. ALONE.
        JSR R5,XOUTL
        DEC R3                  ;NON-PRINTING DELAY FOR CR.
        BR .+6          ;->
RLF:    MOV CCR,R3      ; I     ;EXCLAMATION POINT = CR,LF.
        PRINTC          ;<-
SK4:    GETC                    ;MOVE TO NEXT CHARACTER.
        BR TASK
```

```
;XNSC (GETNSC - GET NON-SPACE CHARACTER) AND XSPNOR (SPNOR - SCAN FOR
;SPACES) BOTH CALL GETC AND BOTH USE REGISTERS R5,R5.

:   GETC
JOR: CMP #40,CHAR                ;IS CHAR A SPACE?
     BNE .+6             ;->
     GETC                ; I     ;YES. GET NEXT CHARACTER.
     BR XSPNOR           ; I     ;TEST IT.
     RTS R5              ;<-     ;NO.

;TESTN - XTESTN TESTS WHETHER THE CHARACTER IN R5 IS
;A PERIOD (1ST RETURN), A LETTER (2ND RETURN),OR A
;NUMBER (3RD RETURN). TESTN USES REGISTERS R0,R2,R3 (CHAR),
;AND R5.

STN: CMP #'.,CHAR                ;IS CHAR A PERIOD?
     BEQ .+4             ;->
     BUMP R5             ; I     ;NO.
     MOV CHAR,SORTCN     ;<-     ;YES - 1ST RETURN.
     SUB #60,SORTCN              ;IS CHAR A LETTER, A #,ETC.?
     BGE .+4             ;->
     RTS R5              ; I     ;NO.
     MOV #-11,R0         ;<-     ;YES.
     ADD SORTCN,R0               ;IS CHAR A #?
     BGT .+4             ;->
     BUMP R5             ; I     ;YES - 3RD RETURN.
     RTS R5              ;<-     ;NO - 2ND RETURN.

;FINDLN - XFIND FINDS A PARTICULAR LINE NUMBER IN THE
;TEXT. UPON ENTRYTHE DESIRED LINE NUMBER RESIDES IN
;"LINENO". FINDLN USES REGISTERS R0,R1,R2,R4,R5. R1
;USED AS LASTLN, R0 USED AS THISLN. AXOUT = R4. UPON
;EXIT R0,THISLN,XPC, AND R4 CONTAIN THE ADDRESS OF THE
;LINE WHOSE LINE # IS > OR = TO THE DESIRED LINE NUMBER.
;R1 AND LASTLN CONTAIN THE ADDRESS OF THE LAST LINE
;TESTED.

IND: MOV CFRS,R1                 ;INITIALIZE POINTERS TO 1ST
     MOV R1,R0                   ;LINE OF TEXT.
     MOV LINENO,R2
NDN: CMP R2,2(R0)                ;IS LINENO >OR= LINE# IN TEXT?
     BGE FEND3-4         ;->     ;YES - SKIP NEXT 4 LINES.
     MOV R0,R1           ; I     ;MOV THISLN TO LASTLN.
     MOV @R0,R0          ; I     ;END OF TEXT?
     BNE FINDN           ; I     ;NO - GOTO FINDN.
     BR FEND3            ; I     ;YES.
     BNE FEND3           ;<-     ;MOVED PAST LINENO-1ST RETURN.
     BUMP R5                     ;FOUND LINENO - 2ND RETURN.
ND3: MOV R1,LASTLN               ;SET UP LASTLN.
     MOV R0,THISLN               ;SET UP THISLN.
     MOV R0,AXOUT                ;SET "TEXTP".
     MOV R0,AXOUT
     AXOUT=R4
     CLR XCT
     RTS R5
```

```
;GETARG - UPON ENTRY R3 SHOULD CONTAIN AN ALPHABETIC
;CHARACTER. IF NOT, AN ERROR DIAGNOSTIC IS GIVEN. IF
;THE VARIABLE BEING SOUGHT IS NOT FOUND, THEN IT IS
;CREATED AND ASSIGNED A VALUE OF ZERO. UPON EXIT THE
;ADDRESS OF THE FOUND VARIABLE (OR CREATED VARIABLE) IS
;STORED IN PT1.
;GETARG USES ALL OF THE REGISTERS.
;GETARG CALLS TESTC,PACKC,GETC,SORTC,TSTLPR,ECALL,SORTV.

ARG  TESTC                  ;TEST CHARACTER.
     ERROR                  ;T.
     ERROR                  ;N.
     ERROR                  ;F.
VAR: CLR ADD1               ;ALPHABETIC CHARACTER-PACK IT
     PACKC                  ;IN 1ST BYTE OF ADD1.
     GETC                   ;GET A CHARACTER.
     SORTC                  ;IS CHARACTER A TERMINATOR?
      .WORD TERMS
     BR GSERCH              ;YES.
     MOVB CHAR,ADD1+1       ;NO - SAVE 2ND LETTER
     GETC            ;<-    ;IN 2ND BYTE OF ADD1.
     SORTC           ; I    ;IGNORE THE REST.
      .WORD TERMS    ; I
     BR GSERCH       ; I
     BR .-8          ;->

RCH: TSTLPR                 ;LOOK FOR SUBSCRIPT VIA SORTCN.
     BR GS1                 ;NOT SUBSCRIPTED BY L-PAR.
     MOV ADD1,R1
     JSR PC,ECALL
     BVS GETARG+4           ;TEST PAREN MATCH.
     POP ADD1               ;RESTORE NAME.
     MOVSTK                 ;PUT THE NUMBER ON THE STACK.
     FIX                    ;CONVERT TO 16-BIT NUMBER
     CMP (R6),R1            ;AND PLACE THE NUMBER IN R1.
     ADD #6,R6              ;DELETE NUMBER ON STACK.
     JSR R5,INTEGER         ;CONVERT TO 16-BIT NUMBER.
:    MOV R1,SUBS            ;STORE SUBSCRIPT.
     PUSH R4
     MOV ADD1,CHAR          ;VARIABLE NAME IN CHAR.
     SORTV                  ;SORT NAME AGAINST SPECIAL LIST1.
TH1:  .BYTE 0,2
      .WORD SPLIST1         ;ADDRESS OF SPECIAL LIST1.
     JMP SPECL1             ;MATCH.
     SORTV                  ;NO MATCH-SORT AGAINST
TH2:  .BYTE 0,2             ;SPECIAL LIST2.
      .WORD SPLIST2         ;ADDRESS OF SPECIAL LIST2.
     JMP SPECL2             ;MATCH.
     SORTV                  ;NO MATCH - SORT NAME
TH3:  .BYTE 0,12            ;AGAINST VARIABLE LIST.
RLST:  .WORD 0
     SKP2            ;->    ;MATCH - SKIP NEXT LINE.
     JMP CREATE      ; I    ;NO MATCH.
RSRH: BUMP ADDRSS    ;<-    ;GET VARIABLE SBSCRPT ADDRESS.
     PUSH ADDRSS            ;SAVE THIS ADDRESS.
     MOV SUBS,CHAR          ;PLACE SUBSCRIPT IN CHAR.
```

```
        JSR R5,STV1             ;SORT SUBS AGAINST VARIABLE LIST.
        SKP2            ;->     ;MATCH - SKIP NEXT LINE.
        JMP CREATE      ; I     ;NO MATCH.
        CMP (R6)+,ADDRSS;<-     ;DOES PUSHED VARIABLE ADDRESS =
        BEQ VFOUND              ;CURRENT ADDRESS? YES-GOTO VFOUND.
        SUB #2,ADDRSS           ;NO - TEST VARIABLE NAME.
        PUSH R2                 ;SAVE NAME ADDRESS.
        MOV ADD1,CHAR           ;VARIABLE NAME IN CHAR.
        JSR R5,STV1             ;IS NAME IN REST OF LIST?
        SKP2            ;->     ;YES - SKIP NEXT LINE.
        JMP CREATE      ; I     ;NO.
        CMP (R6)+,ADDRSS;<-     ;DOES OLD NAME ADDRESS=CURRENT
        BNE VARSRH              ;ONE? NO - GOTO VARSRH.
        BUMP ADDRSS             ;YES - FOUND VARIABLE.
UND:    BUMP ADDRSS             ;SET POINTER.
        MOV ADDRSS,PT1
;       POP R4
        POPJ

        ;SORTV - XSORTV INSTIGATES A SEARCH ON A LIST. UPON
        ;ENTRY THE ENTITY BEING SEARCHED FOR IS IN CHAR. THE
        ;CALLING SEQUENCE FOR SORTV IS
        ;SORTV
        ;  .BYTE LENGTH,VARBYTES         ;NO. OF BYTES IN LIST
                                ;AND THE NO. OF BYTES/VARIABLE.
        ;XXX                    ;RETURN HERE IF MATCH OCCURS.
        ;YYY                    ;RETURN HERE IF NO MATCH OCCURS.
        ;UPON EXIT IF THE ENTITY WAS FOUND, R2 CONTAINS ITS
        ;ADDRESS.

        LSLNG=R1                ;LSLNG=BYTE-LENGTH OF LIST.
        VARBYT=R0               ;VARBYT=#BYTES/VARIABLE
        ADDRSS=R2               ;ADDRSS=ADDRESS IN LIST

ORTV:   MOVB (R5)+,LSLNG
        MOVB (R5)+,VARBYT
        MOVB (R5)+,ADDRSS       ;BEGINNING ADDRESS OF LIST.
        ADD ADDRSS,LSLNG        ;LSLNG=LAST LIST ADDRESS+1
V1:     CMP CHAR,(ADDRSS)       ;DOES CHAR=VARIABLE IN LIST?
        BEQ STVXIT              ;YES - 1ST RETURN.
        ADD VARBYT,ADDRSS       ;NO-GET ADDRESS OF NEXT VAR.
        CMP ADDRSS, LSLNG       ;AT END OF LIST?
        BLT STV1                ;NO - GOTO STV1.
        BUMP R5                 ;YES - 2ND RETURN.
VXIT:   POPJ
```

```
      ;TSTLPR = LPRTST. IF A LEFT PARENTHESIS WASENCOUN
      ;TERED, THEN A MATCH IN THE TERMINATOR LIST OCCURRED
      ;BETWEEN SORTCN = 5 AND 11. TSTLPR TESTS TO SEE IF
      ;IF THE VALUE OF SORTCN IS IN THIS RANGE.

TST   CMP #11,SORTCN          ;IS SORTCN < 11?
      BGE LPRXIT              ;NO - 1ST RETURN.
      CMP #5,SORTCN           ;YES - IS SORTCN > 5?
      BLE LPRXIT              ;NO - 1ST RETURN.
      BUMP R5                 ;YES. L-PAR FOUND - 2ND
XIT:  POPJ                    ;RETURN.
      ;TSTGRP TESTS TO SEE THAT THE LINE IS IN THE GROUP Ø
      ;SPECIFIED BY LINENO. ADDRESS OF A LINEØ IS IN R1.
      ;COMPARE GROUPØ TO THE GROUP NUMBER OF LINENO.

GRP:  MOV (R1),R1
      SWAB R1
      CMPB R1,LINENO+1        ;DO GROUP NUMBERS AGREE?
      BNE .+4           ;=>
      BUMP R5  ; I      ;YES - 2ND RETURN.
      POPJ              ;<-   ;NO - 1ST RETURN.
```

```
$XGETLN FORMULATES THE LINE NUMBER (E.G., IN THE DO AND GOTO
$COMMANDS) AND PLACES THIS NUMBER IN LINENO. THE LOW-
$ORDER BYTE OF LINENO IS THE STEP NUMBER AND THE HIGH-
$ORDER BYTE IS THE GROUP NUMBER. XGETLN GIVES AN ERROR IF
$THE LINE NUMBER IS NOT IN THE RANGE 1.01 TO 99.99.
$(EXCEPTION: DO ALL. THERE LINENO=0 AND NO ERROR DIAG-
$NOSTIC IS GIVEN). XGETLN SETS NAGSV: 000000 FOR A GROUP 0,
$100000 FOR A LINE 0, AND 000001 FOR ALL.
$XGETLN USES REGISTERS R0,R1,R6,R7.
$XGETLN CALLS SPNOR,EVAL, FLOATING POINT PACKAGE(FIX,
$FLT
```

```
                    U_LF),GETC,SORTC.
;MULX=NORMALIZED,FLOATING-POINT 144 (OCTAL)
;FLAC,FLAC1,FLARG ARE INITIAL ADDRESSES OF
;3-WORD LOCATIONS USED FOR STORING AND MANIPULATING
;FLOATING-POINT DATA.
;UPON ENTRY: CHAR IS EITHER A SPACE OR THE 1ST CHARAC-
;TER OF AN ARGUMENT, R4 POINTS TO THE NEXT CHARAC-
;TER OF TEXT. UPON EXIT: CHAR CONTAINS A TERMINATOR
;(",", , ; , CR,↑K), AND R4 POINTS TO THE NEXT CHARACTER
;IN TEXT.

TLN: SPNOR                      ;IGNORE SPACES.
     CLR LINENO                 ;SET LINENO&NAGSW=0
     CLR NAGSW
     CMP #'A,CHAR               ;IS CHAR AN A?
     BEQ GEXIT                  ;YES - GOTO GEXIT.
     MOV CHAR,INSUB             ;NO - SET INSUB<>0.
     JSR PC,EVAL00-2            ;EVALUATE # THAT FOLLOWS
     MOV #FLAC,R0               ;COMMAND KEYWORD AS FLTING-PT.
     MOVTOR0
     FIX=EMT+11
     FIX                        ;CONVERT TO FIXED-PT.(INTEGER)
     MOV FLAC,FLAC1
     BVC .+4            ;->
     ERROR              ; I     ;NUMBER TOO LARGE OR TOO SMALL.
     MOV #FLAC,R1       ;<-     ;SAVE ORIGINAL FLTING-PT #.
     MOV #FLARG,R0
     MOVF+FAM
     PUSHJ TESTA                ;TEST GROUP# FOR CORRECT RANGE.
     FLT=EMT+13
     CMP FLAC1,FLAC             ;SETUP STEP NUMBER IN LOW-
     SUBF=EMT+2                 ;ORDER BYTE OF THE WORD LINENO.
     MOV #FLAC,R1
     MOV #FLARG,R0
     SUBF+FAM
     MULF=EMT+4
     MOV #MULX,R1
     MOV #FLAC,R0
     MULF+FAM
     FIX
     MOV FLAC,FLAC1
     PUSHJ TESTA
```

```
      GETC                        ;TEST FOR TERMINATOR.
      SORTC
       .WORD TLIST
      BR GEXITA
      ERROR             ; I       ;INCORRECT TERMINATOR.
IT:   GETC                        ;MOVE TO TERMINATOR IN THE
      SORTC                       ;DO ALL COMMAND.
       .WORD TLIST
      BR GEXITA
      BR GEXIT
ITA:  TSTB LINENO       ;<-       ;SETUP NAGSW.
      BEQ .+4           ;->
      MOV #100000,NAGSW ;I        ;LINE: NAGSW=100000.
      TSTB LINENO+1     ;<-
      BNE .+4           ;->
      MOV #1,NAGSW      ; I       ;ALL: NAGSW=000001
      POPJ              ;<-       ;GROUP: NAGSW=000000.
TA:   CMP #143,FLAC1
      BLE .+4           ;->
      ERROR             ; I       ;NUMBER IS OUTSIDE RANGE.
      TST FLAC1         ;<-
      BGT .+4           ;-->
      ERROR             ;  I      ;NUMBER IS OUTSIDE RANGE.
      MOVB FLAC1,LINENO ;<-
      POPJ
```

```
      SP=R6
LL:   MOV (SP)+,-4(SP)        ;MOVE SUBROUTINE RETURN DOWN
      PUSH R1                 ;THE PDLIST PAST SAVED CONTENTS
      TST -(SP)               ;OF R1. PUSH RETURN ADDRESS ON
      DEC R4                  ;STACK. BACK UP TEXTPOINTER.

      ;EVAL - EVAL00, EVALUATE AN ARITHMETIC EXPRESSION.
      ;UPON ENTRY R4 POINTS TO CURRENT TEXT POSITION ("(").
      ;ON  EXIT R1,R2,AND R3 CONTAIN THE NUMERIC VALUE OF THE
      ;EXPRESSION. REGISTERS USED - ALL.

L00:  CLR XR5                 ;CLEAR PAREN COUNT.
      PUSH R5                 ;SAVE CONTENTS OF R5.
      MOV #-1,-(SP)           ;PUSH NULL (-1) ON STACK.
L02:  GETNSC                  ;GET A NONSPACE CHARACTER.
      CMP R3,#'+              ;IS IT A "+"?
      BEQ EVAL03              ;YES-IGNORE IT. GOTO EVAL03.
      CMP R3,#'-              ;IS CHARACTER A "-"?
      BNE EVAL01              ;NO - GOTO EVAL01.
      MOV R3,R0               ;YES - SET OPERAND2 = 0.
      CLR R1
      CLR R2
      CLR R3
      BR EVAL05
L03:  GETNSC                  ;GET A CHARACTER.
L01:  CMP R3,#'(              ;IS IT AN OPEN PAREN?
      BNE EVAL4               ;NO - GOTO EVAL04.
      CLR -(SP)               ;PUSH A ZERO ON THE STACK.
      INC XR5                 ;INCREMENT PAREN COUNT.
      BR EVAL02               ;GO BACK TO EVAL02.
AL04: DEC R4                  ;MOVE CHARAC POINTER BACK ONE.
      PUSH XR5                ;STORE PAREN COUNT
      PUSH YR5                ;AND OTHER DATA.
      PUSH R5
      JSR PC,GTP00            ;GET AN OPERAND.
      POP R5                  ;RESTORE DATA.
      POP YR5
      POP XR5
AL12: GTOPR                   ;GET AN OPERATOR (IN R0).
      TST @SP                 ;IS THE STACK NULL?
      BGT EVAL06              ;NO - GOTO EVAL06.
      BR EVAL17
AL05: MOVSTK                  ;PUSH OPERAND ON THE STACK.
      MOV R0,-(SP)            ;PUSH OPERATOR ON STACK.
      BR EVAL03               ;GO BACK TO EVAL03.
AL06: PUSH R4                 ;SAVE THE TEXTPOINTER.
      MOV #EVAL07+7,R4        ;GET THE TABLE ADDRESS.
AL08: CMPB -(R4),R0           ;FIND OPERATOR2.
      BNE EVAL08              ;IT MUST BE FOUND.
      ASR R4                  ;GET RID OF BYTE POINTER.
      MOV R4,R5               ;STORE IN R5.
      MOV #EVAL07+7,R4        ;GET TABLE ADDRESS.
AL09: CMPB -(R4),2(SP)        ;FIND OPERATOR1.
      BNE EVAL09              ;IT MUST BE FOUND.
      ASR R4                  ;CLEAR LOW-ORDER BIT.
      MOV R4,YR5              ;SAVE IT FOR NOW.
      MOV (SP)+,R4            ;RESTORE TEXTPOINTER.
      CMP YR5,R5              ;IS PRIORITY OF OPERATOR1>OPERATOR2?
      BLT EVAL05              ;NO - GO VACK TO EVAL05.
      MOV R0,R5               ;YES-SAVE OPERATOR2.
      MOV #EVAL07+7,R0        ;GET TABLE ADDRESS.
```

```
L10: CMPB -(R0),@SP      ;FIND OPERATOR.
     BNE EVAL10          ;IT MUST BE FOUND.
     SUB #EVAL07+2,R0    ;GET DISPLACEMENT.
     ASL R0
     ADD #EVAL11,R0      ;GET ROUTINE ADDRESS.
     MOV R0,XR5          ;SAVE IT.
     TST (SP)+           ;DISCARD OLD OPERATOR1.
     MOV SP,R0           ;GET DESTINATION ADDRESS.
     MOVSTK              ;PUT SOURCE ON STACK.
     PUSH R4             ;SAVE TEXTPOINTER.
     MOV SP,R1           ;GET
     BUMP R1             ;SOURCE ADDRESS AND
     MOV XR5,R3          ;ROUTINE ADDRESS.
     PUSH R5             ;SAVE OPERATOR2.
     JSR PC,0(R3)        ;GO COMPUTE VALUE.
     POP R0              ;RESTORE OPERATOR2.
     POP R4              ;RESTORE TEXTPOINTER.
     ADD #6,SP           ;DISCARD SOURCE.
     POP R1              ;PLACE RESULT
     POP R2              ;IN
     POP R3              ;OPERAND2.
     TST @SP             ;IS STACK NULL?
     BGT EVAL06          ;NO-TAKE CARE OF REST OF STACK.
L17: CMP R0,#')          ;IS OPERATOR A CLOSED PAREN?
     BEQ EVAL14          ;YES - GOTO EVAL14.
     TST R0              ;NO. IS IT A NULL?
     BGT EVAL05          ;NO - GOTO EVAL05.
     TST XR5             ;IS THE PAREN COUNT ZERO?
     BNE EVAL13          ;NO - GOTO EVAL13.
     BUMP SP             ;YES. POP NULL.
     POP R5              ;RESTORE CONTENTS OF R5.
     CCC                 ;CLEAR CONDITION CODES.
     RTS PC              ;RETURN WITH RESULT IN R1,R2,R3.
L13: ERROR               ;PAREN COUNT BAD.
L14: TST XR5             ;IS PAREN COUNT ZERO?
     BNE EVAL15          ;NO - GOTO EVAL15.
L16: BUMP SP             ;POP NULL.
     POP R5              ;RESTORE CONTENTS OF R5.
     SEV                 ;YES - ERROR MESSAGE LATER.
     RTS PC
L15: TST @SP             ;JUMP
     BLT EVAL16          ;IF STACK = -1.
     BUMP SP             ;POP NULL OFF STACK.
     DEC XR5             ;DECREASE PAREN COUNT BY ONE.
     BR EVAL12

AL07:  .BYTE 0,#')       ;DO NOT
       .BYTE #'+,#'-     ;CHANGE
       .BYTE #'*,#'/     ;THE ORDER OF
       .BYTE #'↑         ;THIS TABLE
       .EVEN
AL11: ADDF00             ;THIS TABLE
      SUBF00             ;GOES WITH THE
      MULF00             ;ONE ABOVE -
      DIVF00             ;SO DO NOT CHANGE
      PWRF00             ;ORDER.
```

```
      ;GTOPR - GTPR00, GET AN OPERATOR. LOOK FOR +,-,*,/,
      ;↑, AND ). IF FOUND R0 RETURNS OPERATOR AND R1 POINTS
      ;TO NEXT CHARACTER. IF NOT FOUND, R0 RETURNS ZERO AND
      ;R1 POINTS  TO CHARACTER WHERE FAILURE OCCURRED.
      ;REGISTERS USED - R0,R3,R4.

R00:  PUSH R3                    ;SAVE CHARACTER.
      GETNSC                     ;GET NONSPACE CHARACTER.
      MOV #EVAL07+7,R0           ;GET ADDRESS OF LIST.
R01:  CMPB -(R0),R3              ;IS CHARACTER A LEGAL OPERATOR?
      BEQ GTPR02                 ;YES - GOTO GTPR02.
      CMP R0,#EVAL7+1 ;HAS SEARCH FAILED?
      BGT GTPR01                 ;NO.
      CLR R0                     ;YES - SET R0=0 AND BACKUP
      DEC R4                     ;POINTER TO PT. OF FAILURE.
R03:  POP R3                     ;RESTORE CHARACTER.
      RTS PC
R02:  MOV R3,R0                  ;PUT CHARACTER IN R0.
      BR GTPR03

      ;MOVSTK - MOVS00,MOVE REGISTERS R3,R2,R1 ONTO THE
      ;STACK.
S00:  PUSH R3
      PUSH R2
      PUSH R1
      RTS PC

      ;GETOP - GTP00, GET AN OPERAND.
      ;UPON ENTRY R4 POINTS TO THE START OF AN OPERAND.
      ;UPON EXIT R1,R2,R3 CONTAIN THE VALUE OF THE OPERAND
      ;IF LEGAL. IF NOT LEGAL, A FATAL ERROR CALL IS MADE.
      ;ON LEGAL OR ILLEGAL EXITS, R4 WILL ALWAYS POINT
      ;ONE CHARACTER AFTER THE SCAN WAS ENDED. NOTE:
 BE R[illegible]IGENCE IT MAY, BY WAY
      ;OF CALLS TO "EVAL", RE-ENTER ITSELF BEFORE
      ;COMPLETION. REGISTERS USED - ALL.

00:   PUSH R4                    ;SAVE TEXTPOINTER.
      GETNSC                     ;GET NONSPACE CHARACTER.
      TESTC                      ;THE CHARACTER IS A
      BR GTP15                   ;TERMINATOR-GOTO GTP15.
      BR GTP18                   ;NUMERIC.
      JMP EFUN                   ;FUNCTION.
      BR GTP02                   ;ALPHABETIC.
18:   POP R4                     ;RESTORE TEXTPOINTER.
      SUB #6,SP                  ;RESERVE SPACE FOR #.
      MOV SP,R0                  ;MOVE ADDRESS FOR # TO R0.
      ATOF                       ;ASCII TO NUMERIC CONVERSION.
      POP R1                     ;PLACE THE #
      POP R2                     ;IN R1,R2,R3.
      POP R3
      RTS PC
```

```
01:  POP R1                    ;GET THE
     POP R2                    ;NUMBER
     POP R3                    ;AND
     RTS PC                    ;RETURN.
15:  CMP R3,#'.                ;DOES # BEGIN WITH DECIMAL PT?
     BEQ GTP18                 ;YES - GOTO GTP18.
     BR GTP12                  ;NO.
02:  GETARG                    ;PLACE VARIABLE ADDRESS IN PT1.
     MOV PT1,R0                ;PLACE VARIABLE ADDRESS IN R0.
     MOV (R0)+,R1
     MOV (R0)+,R2
     MOV (R0),R3
     ;NOTE THAT IF VARIABLE DOES NOT EXIST, THEN GETARG
     ;WILL CREATE THE VARIABLE AND SET IT = 0.
N:   CLR R1                    ;CLEAR CODE STORAGE.
N1:  GETC                      ;READ FUNCTION NAME.
     SORTC                     ;THE F (1,2,OR 3 LETTERS).
      .WORD TERMS              ;IS CHARAC A TERMINATOR?
     BR EFUN2                  ;YES.
     ASL R1                    ;NO. TRANSLATE CHARACTER
     ADD CHAR,R1               ;TO FUNCTION CODENAME.
     BR EFUN

UN2: CMP R1,#500               ;COMPLETE TRANSLATION TO
     BLT .+6          ;->      ;FUNCTION CODENAME.
     SUB #500,R1      ; I
     AND #000377,R1   ;<-
     MOV R1,EFOP               ;SAVE FUNCTION CODENAME.
     TSTLPR                    ;WAS TERMINATOR A L-PAREN?
     ERROR                     ;NO.
     JSR PC,ECALL              ;YES. EVALUATE FCN ARGUMENT.
     POP EFOP                  ;RESTORE FUNCTION CODENAME.
     MOVSTK                    ;MOVE FCN ARGUMENT TO STACK.
     MOV EFOP,R3
     SORTC                     ;IS FUNCTION CODENAME
      .WORD FNTABL             ;IN LIST?
     SKP2                      ;YES- SKIP NEXT LINE.
     ERROR                     ;NO.
     ADD #FNTABF,SORTCN        ;GET ADDRESS OF POINTER.
     MOV SP,R1                 ;SOURCE ADDRESS IN R1.
     PUSH R4                   ;SAVE CHARACTER POINTER.
     SUB #6,SP                 ;RESERVE SPACE ON STACK.
     MOV SP,R0                 ;DESTINATION ADDRESS.
     JSR PC,@(R2)              ;GO EXECUTE FUNCTION.
     POP R1                    ;MOVE ANSWER
     POP R2                    ;TO REGISTERS R1,R2, AND
     POP R3                    ;R3.
     POP R4                    ;RESTORE TEXTPOINTER.
     ADD #10,SP                ;DISCARD USELESS DATA ON STACK.
     RTS PC
TP12: ERROR                    ;INCORRECT ARGUMENT.
```

```
ST:   .BYTE 40,',,'.,15,13,200        ;7-BIT ANSCII CHARACTERS -
      ;SPACE, ",", "., CR,↑K,NEG. NO. TO END LIST.

      TLIST=GLIST+1
      TLISTA=GLIST+2
      TLISTE=GLIST+3

LIST: .BYTE 'S,'F,'I,'G,'O,'D,'C,'A,'T,'L     ;1ST LETTERS OF
      ;COMMANDS - SET,FOR,IF,GOTO,ON,DO,COMMENT,ASK,TYPE,LIBRARY.

       .BYTE 'E,'W,'M,'Q,'R,'*,200
      ; ERASE,WRITE,MODIFY,QUIT,RETURN,ADDITIONAL SPACE FOR
      ;USER-DEFINED COMMANDS. NEG. NO. ENDS LIST.

      .EVEN

GO:   SET
      FOR
DD:   IF
      GOTO
      ON
      DO
      COMMENT
      ASK
      TYPE
      LIBRARY
      ERASE
      WRITE
      MODIFY
      QUIT
      RETURN
      *                               ;* - PLACE ADDED COMMAND HERE.

NT:   CRPRNT                          ;CR - PRINT CR,LF.
      CKPRNT                          ;↑K - PRINT ↑,K,CR,LF,6 SPACES.


PLS:  .BYTE '(,'[
IST:  .BYTE '<,'=,'>,200              ;NEG. NO. ENDS LIST.

OP:   AA                              ;(
      AA                              ;[
      AB                              ;<
      AC                              ;=
      AD                              ;>
```

```
ST:   IF1                       ;"," "
      PROCESS                   ;"," "
      PC1                       ;CR
      PROCESS                   ;↑K (CONTROL/K)

MS:     .BYTE 40,'+,'-,'/,'*,'↑          ;SPACE,"+","-","/","*","↑".
        .BYTE '(,'[,'<,'),],'>           ;"(","[","<",")","]",">".
        .BYTE ',,',;,15,13,'=,200        ;",",",";",CR,↑K,"=,NEG.NO.

ST1:  FINCR                     ;"," = STANDARD FORMAT.
      PROCESS                   ;";" = SET,PLUS....
      PC1                       ;CR = SET COMMAND.
      PROCESS                   ;↑K = SET COMMAND.
ST2:  FLIMIT                    ;"," = STANDARD.
      FINFIN                    ;";" = SHORT
      ERROR                     ;CR
      FINFIN                    ;↑K

IST:    .BYTE '%,'",'!,'#,'$,200         ;NEG.NO. ENDS LIST.


LIST: TINTR                     ;% - FORMAT DELIMITER.
      TQUOT                     ;" - LITERAL DELIMITER.
      TCRLF                     ;! - CR AND LINE FEED.
      TCRLF2                    ;# - CR ONLY.
      TDUMP                     ;$ - DUMP THE SYMBOL TABLE CONTENTS.
      TASK4                     ;SPACE - TERMINATOR FOR NAMES.
      TASK4                     ;, - TERMINATOR FOR EXPRESSIONS.
      PROCESS                   ;"," - TERMINATOR FOR COMANDS.
      PC1                       ;CR - TERMINATOR FOR STRINGS.
      PROCESS                   ;↑K- TERMINATOR FOR COMMANDS.

IST2:   .BYTE '",15,200         ;LIST CONSISTS OF " AND CR.
                                ;NEG.NO. ENDS LIST.

      .EVEN

IST3: TASK4                     ;"
      PC1                       ;CR = AUTOMATIC QUOTE SWITCH.

      ;CODES FOR FUNCTIONS (LISTED WITHOUT AN INITIAL F).

NTABL:  .BYTE 33,2,150,136              ;ABS,SQT,SGN,ITR
        .BYTE 115,126,124,154           ;RAC,MOD,EXP,SIN
        .BYTE 75,72,125,44              ;COS ATN,LOG,AND
        .BYTE 160,20,67,113             ;OR XOR,COM EQU
        .BYTE 17,175,157,53             ;ADC,SW,TIM,DAY
        .BYTE 130,117,101               ;RAN,LIM,ILE.

      .EVEN
```

```
;ADDRESSES OF FUNCTIONS.

TABF:
                        ;FABS
                        ;FSQT
                        ;FSGN
                        ;FITR
                        ;FRAC
                        ;FMOD
                        ;FEXP
                        ;FSIN
                        ;FCOS
                        ;FATN
                        ;FLOG
                        ;FAND
        ERROR           ;FOR
        ERROR           ;FXOR
        ERROR           ;FCOM
        ERROR           ;FEQU
        ERROR           ;FADC
        ERROR           ;FSU
        ERROR           ;FTIM
        ERROR           ;FDAY
        ERROR           ;FRAN
        ERROR           ;FLIM
        ERROR           ;FILE
        ERROR

        .EVEN
```