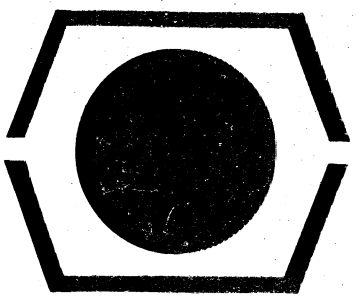


DECUS
PROGRAM LIBRARY

DECUS NO.	FOCAL8-148
TITLE	FOCL.S, AN EXPANDED LANGUAGE FOR SMALL COMPUTERS, BASED ON FOCAL
AUTHOR	D. E. Wrege
COMPANY	Submitted by: John Alderman Georgia Tech Atlanta, Georgia
DATE	November 28, 1970
SOURCE LANGUAGE	PAL 8

Although this program has been tested by the contributor, no warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related program material, and no responsibility is assumed by these parties in connection therewith.

FOCL.S, AN EXPANDED LANGUAGE FOR SMALL COMPUTERS, BASED ON FOCAL

DECUS Program Library Write-up

DECUS NO. FOCAL8-148

DESCRIPTION

FOCL/S IS A ONE USER VERSION OF FOCAL/69 FOR 8K PDP-8/I2 COMPUTERS. AMONG THE ADDITIONAL FEATURES ARE:

-UP TO 8 MULTIPLE SUBSCRIPTED ARRAY VARIABLES.

-TEXT, VARIABLES, PUSHDOWN LIST, AND EXTENDED FUNCTIONS HAVE BEEN MOVED TO FIELD 1 THUS : (1) FREEING CORE IN FIELD Ø FOR USER DEFINED FUNCTIONS. (2) MORE EFFICIENT TRADE OFF BETWEEN TEXT STORAGE AND VARIABLES.

-XOD COMPATIBILITY FOR EASY DEBUGGING.

-DIRECT ACCESSING OF CORE.*

-EXECUTION OF MACHINE LANGUAGE INSTRUCTIONS, THUS ALLOWING ABILITY OF DRIVING SPECIALIZED HARDWARE DIRECTLY WITH FOCL/S.*

-LOGICAL 'AND' BETWEEN EXPRESSIONS.**

-OCTAL TO DECIMAL CONVERSION.

-DECIMAL TO OCTAL CONVERSION.

-'ON' COMMAND. - A THREE BRANCH "CONDITIONAL DO"; CALLS LINES OR GROUPS AS SUBROUTINES UPON CONDITION OF AN EXPRESSION (SIMILAR TO IF ONLY 'DO' IS EXECUTED INSTEAD OF 'GOTO')

-LINE NUMBER COMPUTATION FROM ARITHMETIC EXPRESSIONS.***

-NO LIMIT (EFFECTIVELY) TO LENGTH OF DIRECT COMMAND LINE.

-CONT/K - CONTINUATION CHARACTER FOR TEXT/COMMAND LINES.

-TDUMP TYPES VARIABLES IN ORDER OF MOST RECENTLY CREATED FIRST: AS IN DEBUGGING ONE IS USUALLY INTERESTED IN THE MOST RECENTLY CREATED VARIABLES.

* THANKS TO "FOCAL+" BY D.DYMENT
** THANKS TO "FAND" BY J.C.ALDERMAN
*** THANKS TO A.B.WILSON

COMPATIBILITY WITH FOCAL/69

SYNTACTICALLY FOCL/S IS COMPLETELY COMPATIBLE WITH FOCAL/69. THE PROGRAM WAS INTENDED TO BE A SUPERSET OF FOCAL/69. THE ONLY DIFFERENCES ARE:

-THE LIBRARY EXIT COMMAND HAS BEEN REMOVED: HOWEVER THE USER MAY EXAMINE THE EQUIVALENT LOCATIONS THROUGH USE OF THE FX(1,X,Y) COMMANDS AND CONVERT THEM TO OCTAL THROUGH USE OF THE FX(8,N) COMMAND
-T $ CAUSES TYPEOUT OF THE VARIABLES IN REVERSE ORDER TO THAT IN WHICH THEY WERE CREATED
-THE ERROR DIAGNOSTIC LIST IS EXTENDED AND CHANGED

MULTIPLE SUBSCRIPTED ARRAYS

THESE ARRAYS ARE STORED IN A "COMMON" STORAGE AREA, AVAILABLE TO ALL FOCL/S PROGRAM SEGMENTS. THEY ARE INITIALIZED WITH THE LIBRARY OPEN COMMAND, WHICH HAS THE FOLLOWING SYNTAX:

LIBRARY OPEN, NAME, ORIGIN, LIMIT1, LIM2, LIM3, LIM4 (RETURN)

FOR EXAMPLE:

LIBRARY OPEN,X,Ø,2,2,2,2
WILL OPEN A 4-DIMENSIONAL ARRAY NAMED X OF MAXIMUM LENGTH 2 IN EACH DIMENSION. THE VARIABLE ORIGIN PARAMETER IS Ø, WHICH MEANS THAT THE ARRAY WILL BE STORED AT THE LOWER BOUND (BEGINNING) OF THE VARIABLE STORAGE AREA. IN THE PRESENT VERSION THE LIBRARY OPEN COMMAND MUST BE THE LAST STATEMENT ON A LINE.

VARIABLE STORAGE AREA IS ASSIGNED BY THE LIBRARY LIMIT COMMAND, WHICH HAS THE FOLLOWING SYNTAX.

LIBRARY LIMIT, EXPRSSION

IT IS NECESSARY AT THE BEGINNING OF THE PROGRAM (OR AT LEAST PRIOR TO THE LIBRARY OPEN COMMAND) TO SET THE LIMIT VIA THIS COMMAND IN ORDER TO DEFINE THE BORDER (AT ABSOLUTE ADDRESS 'EXPRESSION') BETWEEN THE COMMON ARRAY STORAGE AREA AND THE VARIABLE/TEXT AREA.
EXPRESSION IS LIMITED TO 1341(OCTAL)<'EXPRESSION'<7577(OCTAL)
-EXAMPLE: L L,4095-128-3*N WILL SET LIMIT TO ALLOW SPACE FOR N VARIABLES
(SEE CORE MAP FOR MORE COMPLETE DESCRIPTION)

THE COMMAND

LIBRARY CLOSE, NAME

WILL RELEASE THE VARIABLE NAME FROM ITS RESERVED ARRAY STATUS.

NOTE THAT ADDRESS COMPUTATION IS CARRIED OUT ON ARRAYED VARIABLES, THUS REQUIRING ONLY 3-WORDS PER VARIABLE. I.E. VARIABLE STORAGE IS MORE EFFICIENT WITH REGARD TO CORE STORAGE THAN 'NORMAL' VARIABLES (5-WORDS)

LIMITATIONS

THERE ARE MAXIMALLY 8 SUCH DIMENSIONED ARRAYS AVAILABLE. EACH MAY HAVE A MAXIMUM DIMENSIONALITY OF UP TO 4 SUBSCRIPTS. SUBSCRIPTS MUST BE GREATER THAN Ø AND WILL BE CONSIDERED INTEGERS. THE SUBSCRIPTS ARE EVALUATED AS THE 12 BIT LOW ORDER INTEGER PART OF 'LIMITN' AND IS ASSUMED POSITIVE.

MAY NOT BE USED TO LEFT OF "=" IN FOR STATEMENT.

THE ONLY CHECK ON THE VALIDITY OF THE ADDRESS COMPUTATION IS THAT THE RESULTANT ADDRESS IS WITHIN THE AVAILABLE ARRAY STORAGE AREA AS SET BY THE LIBRARY LIMITS COMMAND. HENCE IT IS POSSIBLE TO USE SUBSCRIPTS WHICH ARE OUTSIDE OF THE DIMENSIONAL LIMIT - HOWEVER, THE USER MUST BE AWARE THAT OTHER ARRAYED VARIABLES MAY BE OVERWRITTEN. FOR EXAMPLE

```
*L L,3000
*L 0,A,0,2,2,2
*S A(1,1,3)=0
```

THE LAST COMMAND WILL NOT GENERATE AN ERROR MESSAGE - BUT THE VARIABLE A(1,2,1) WILL BE SET EQUAL TO 0 AS THE ADDRESS COMPUTED BY A(1,1,3) IS THE SAME AS THAT OF A(1,2,1).

ARRAY ADDRESS COMPUTATION

THE ARRAYS ARE STORED IN ROW-COLUMN ORDER, AND IT MAY BE USEFUL TO ASSIGN MORE THAN ONE ARRAY TO THE SAME STORAGE AREA SO THAT IT MAY BE ADDRESSED THROUGH A DIFFERENT ADDRESS COMPUTATIONAL ALGORITHM.

SUBSCRIPTS ARE CONSIDERED IN ORDER OF ROW, THEN COLUMN ADDRESS. THE LIMITS MENTIONED ABOVE ARE THE LENGTHS OF THE ROWS. IF THE SUBSCRIPTS ARE CONSIDERED IN THE ORDER OF PRESENTATION AS A(1),A(2), A(3),A(4), AND THE LIMITS ARE CORRESPONDINGLY L(1),L(2),L(3), AND L(4), THEN THE FOLLOWING FORMULA MAY BE USED TO COMPUTE THE RELATIVE ADDRESS (FROM THE BEGINNING OF THE AREA ASSIGNED TO THE ARRAY) OF AN ELEMENT IN THE ARRAY:

```
(((A(1)-1)*L(2)+A(2)-1)*L(3)+A(3)-1)*L(4)+A(4)-1
```

IN THE EXAMPLE, FOR LIMITS 2,2,2,2, THE ABOVE SIMPLIFIES TO:

```
(((A(1)-1)*2+A(2)-1)*2+A(3)-1)*2+A(4)-1
```

THE FOLLOWING FOCL/S PROGRAM WILL PERFORM THE EQUIVALENT COMPUTATION:

```
SET IN=A(1)-1;FOR M=2,4;SET IN=IN*L(M)-1
```

THESE COMPUTATIONS ARE, OF COURSE, PERFORMED BY FOCAL/S AUTOMATICALLY, BUT IT IS USEFUL TO KNOW THE ADDRESSING ALGORITHMS, EXPECIALLY FOR PERFORMING "DUMPS" OF THE ARRAYED STORAGE.

EXAMPLE: TO CREATE TWO VARIABLES WHICH DO NOT OVERLAP:

```
L 0,A,0,2,2
L 0,B,5,4,4
```

IN THIS EXAMPLE THE ORIGIN OF THE "B" VARIABLE WAS SET TO 5 AS THE "A" VARIABLE TAKES UP THE FIRST 4 ARRAY LOCATIONS. HENCE, THROUGH THE USE OF THE ORIGIN VARIABLES MAY BE CREATED IN COMMON OR NOT IN COMMON.

"ON" COMMAND

THE "ON" COMMAND IS A THREE BRANCH CONDITIONAL 'DO'. IT FUNCTIONS SIMILAR TO THE IF COMMAND (ACTUALLY USES SOME OF 'IF'S CODING), EXCEPT THAT THE LINE NUMBER (OR GROUP) IS EXECUTED AS IN A 'DO' COMMAND RATHER THAN AS IN A 'GOTO' COMMAND.

```
--EXAMPLE: ON (X)2.1,5,3.4;D 6;CONTINUE
        IN THIS CASE IF X<0  LINE 2.1 IS EXECUTED THEN GROUP 6
                                IS EXECUTED;CONTROL CONTINUES ON
                                TO NEXT LINE
                     X=0  GROUP 5 IS EXECUTED;THEN GROUP 6
                     X>0  LINE 3.4 IS EXECUTED;THEN GROUP 6
```

LINE NUMBER COMPUTATION

FOCL/S WILL ALLOW ANY ARITHMETIC EXPRESSION FOR A LINE NUMBER IN 'DO', 'IF','GOTO','MODIFY', OR 'ON' COMMANDS.

```
--EXAMPLE: ON (X)-X,4.1,X+.3;C-REST OF LINE
              IN THIS EXAMPLE IF X=-2, GROUP 2 WOULD BE
              EXECUTED AS A SUBROUTINE WITH RETURN TO REST OF LINE.
              IF X=0, LINE 4.1 WOULD BE EXECUTED;THEN REST OF LINE
              IF X=2, LINE 2.3 WOULD BE EXECUTED, WITH CONTROL RETURNED
              TO REST OF LINE.
```

IF THE LINE NUMBER STARTS WITH A NUMBER, THE OLD SCHEME OF LINE NUMBER COMPUTATION IS CARRIED OUT. AS THE LINE NUMBER CALCULATION FROM ARITHMETIC EXPRESSIONS USES THE FLOATING POINT PACKAGE, IT IS RATHER SLOW. HENCE, IN FOR LOOPS ETC, WHERE ONE WISHES TO SAVE TIME, EXPLICIT LINE NUMBER EXPRESSIONS ARE DESIREABLE.

```
--EXAMPLE:  DO X;       TAKES LONGER THAN       DO 3.1
        TO COMPUTE THE LINE NUMBER. SIMILARLY "DO 2*X" IS
        ILLEGAL AS THE EXPRESSION STARTS OUT WITH A NUMBER (2).
        WHEREAS "DO X*2" WILL COMPUTE A LINE NUMBER.
```

CONTINUATION CHARACTER CNT/K

IN ORDER TO IMPLEMENT LONG LINES OF DIRECT OR INDIRECT COMMANDS A SPECIAL CHARACTER (CONTROL/K) IS USED. WITHIN FOCL/S THIS CHARACTER FUNCTIONS THE SAME AS ";" EXCEPT WHENEVER IT IS TO BE TYPED FOCL/S TYPES "↑K(CR)(LF)(6 SPACES)". HENCE MAKING PROGRAM EASY TO READ.

```
-EXAMPLE- 02.10 S X=(X↑2+FSQT(Y+3-Z)+(A(1,J,K)+B(J,K))↑4)↑K
                S X=X/FSIN(THETA)+A(2,J,K)
```

STRICTLY SPEAKING THIS IS NOT A CONTINUATION CHARACTER AS IN FORTRAN AS IT ALSO IS A COMMAND TERMINATOR.

COMMAND STRINGS

FOCL/S HAS NO SEPARATE COMMAND BUFFER FOR DIRECT COMMANDS. IN THE STAR (*) MODE THE COMMAND STRING IS APPENDED TO THE INDIRECT COMMAND BUFFER: HENCE COMMAND STRINGS MAY BE AS LONG AS THE USER LIKES. BY USE OF THE ↑K CHARACTER LONG AND POWERFUL COMMAND STRINGS MAY BE IMPLEMENTED

FUNCTIONS AVAILABLE

SEVERAL FUNCTIONS HAVE NOT BEEN IMPLEMENTED IN FOCL/S AS (1) THE ORIGINAL ROUTINES ARE NOT VERY GOOD, OR (2) I HAVE FOUND THAT USERS PREFER TO CODE THEM FOR THEIR SPECIFIC HARDWARE. THE FUNCTIONS THAT HAVE BEEN REMOVED ARE:

FRAN
FADC

THE FX FUNCTION HAS BEEN EXPANDED TO PERFORM SEVERAL DIFFERENT OPERATIONS AND TO ALLOW EXPANSION FOR USER DEFINED FUNCTIONS. THE GENERAL FORM IS:

FX(N,ARG1,ARG2,---)

WHERE N IS PRESENTLY 1,2,3,8,10 (DECIMAL).

FX(1,ARG1,ARG2)

THE CORE MEMORY FUNCTION (FCOR IN FOCAL+ BY D.DYMENT). THIS FUNCTION MAY BE USED IN TWO DISTINCT WAYS; FX(1,ARG1) TAKES AS ITS VALUE THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY 'ARG1' (WHICH MUST BE A DECIMAL VALUE IN THE ANGE 0-32767); FX(1,ARG1,ARG2) PERFORMS SIMILARLY, FIRST DEPOSITING 'ARG2' IN THE SPECIFIED MEMORY LOCATION. THUS THE STATEMENT "SET X=FX(1,12345,FX(1,12345)+1)", WOULD INCREMENT THE CONTENTS OF MEMORY LOCATION 12345 (FIELD 3, LOCATION 0071(OCTAL)) AND SET 'X' EQUAL TO THE NEW VALUE.
ALL ARGUMENTS ARE DECIMAL - THE FX(10,-)AND FX(8,-) MAY BE USED TO CONVERT FROM OCTAL TO DECIMAL.

FX(2,ARG1,ARG2)

THE "EXECUTE" FUNCTION (FXCT IN FOCAL+ BY D.DYMENT). THIS FUNCTION WHICH TAKES TWO ARGUMENTS - FX(2,ARG1,ARG2) WILL EXECUTE THE MACHINE LANGUAGE INSTRUCTION SPECIFIED BY 'ARG1'. THE VALUE OF 'ARG2' WILL BE PLACED IN THE ACCUMULATOR PRIOR TO THIS INSTRUCTION EXECUTION, AND THE VALUE OF THE FUNCTION IS THAT SPECIFIED BY THE ACCUMULATOR FOLLOWING INSTRUCTION EXECUTION. BOTH ARGUMENTS MUST BE DECIMAL VALUES IN THE RANGE OF 0-4096; 'ARG2' IS ASSUMED TO BE ZERO IF OMITTED. NEEDLESS TO SAY, "JMP" INSTRUCTIONS AND THE LIKE SHOULD BE EXECUTED ONLY WITH EXTREME CAUTION. A SIMPLE EXAMPLE -- SET Y=FX(2,3844) WILL SET Y EQUAL TO THE VALUE OF THE SWITCH REGISTER.

BRANCH

THE BRANCH COMMAND IS A CONDITIONAL "GOTO" COMMAND. THIS COMMAND FUNCTIONS IN A MANNER IDENTICAL TO THE 'GOTO' COMMAND EXCEPT THAT IT WILL FUNCTION AS A NOP IF THE LAST INSTRUCTION EXECUTED BY THE FX(2,--) FUNCTION CAUSED A SKIP TO OCCUR. THUS THE PRESENCE OR ABSENCE OF A SKIP MAY BE TESTED. EXAMPLE --

```
3.14 SET Z=FX(2,3089);BRANCH 3.14
```

THIS WILL CAUSE THE PROGRAM TO WAIT UNTIL THE HIGH SPEED PUNCH FLAG IS SET, AND THEN CONTINUE TO THE NEXT LINE OF INSTRUCTIONS.

NOTE: USERS OF THE FX(2,--) COMMAND TO CONTROL PERIPHERAL HARDWARE MUST KEEP IN MIND THAT FOCL USES THE INTERRUPT SYSTEM FOR I/O SERVICING, AND IS NOT SET UP TO HANDLE INTERRUPTS GENERATED BY ADDITIONAL DEVICES. USERS EXPECTING INTERRUPT REQUEST FLAGS MAY EXTEND THE SKIP CHAIN TO HANDLE FLAGS FOR THE NEW DEVICES (FX(1,--) MAY BE USED FOR THIS).

FX(3,ARG1,ARG2)

THIS IS THE 'AND' FUNCTION (FIRST IMPLEMENTED BY J.C.ALDERMAN). THIS FUNCTION CONVERTS ARG1 AND ARG2 TO THE LOW ORDER 12 BITS OF THEIR INTEGER VALUE, AND THEN PERFORMS THE 'LOGICAL AND' OF THEM, LEAVING THE RESULT AS THE VALUE OF THE FUNCTION. THIS FUNCTION IS USEFUL FOR MASKING OPERATIONS, ESPECIALLY WITH STATES MASKS, ETC., USED WITH EXTERNAL DEVICES. -- EXAMPLE: TYPE FX(3,255,A) WILL PRESERVE ONLY THE LOW ORDER 8 BITS OF A.

FX(8,ARG)

THIS FUNCTION TAKES THE VALUE 'ARG' AND CONVERTS IT TO ITS OCTAL EQUIVALENT. NOTE THAT THE VALUE OF THE FUNCTION IS A DECIMAL-FLOATING POINT NUMBER, HOWEVER, IT WILL BE THE DECIMAL OCTAL-EQUIVALENT OF THE DECIMAL VALUE 'ARG'.
EXAMPLE: FX(8,219) OR FX(8,512) WILL HAVE THE VALUE 1000.

FX(10,ARG)

THIS FUNCTION CONVERTS THE (DECIMAL) OCTAL NUMBER 'ARG' AND CONVERTS IT TO ITS DECIMAL EQUIVALENT. (SEE ABOVE). EXAMPLE: FX(10,2048) WILL HAVE THE VALUE 4000 OR 2↑11.

ADDITIONAL FUNCTIONS

IN ADDITION TO THE LINKS TO THE ABOVE FUNCTIONS, LINKS ARE AVAILABLE TO FX(N,---) FOR N=4,5,6,7 BY ENTRY OF THE APPROPRIATE ENTRY ADDRESS INTO THE TABLE FXGO. (SEE TECHNICAL SECTION TO FOLLOW FOR FURTHER DETAILS)

TECHNICAL DETAILS

EXTENDED FUNCTIONS

HAVE BEEN MOVED TO FIELD 1. OCCUPYING UP TO LOCATION 1340.
ROUTINES MODIFIED OR DELETED

- FPNT
- ALL EXTENDED FUNCTIONS
- FSQT
- LIB - DELETED
- NORMAL FIELD 1 PART OF 8K OVERLAY

INITIAL DIALOGUE

HAS BEEN REMOVED. HOWEVER A DIALOGUE MAY BE TAILOR MADE TO THE USER'S ENVIRONMENT THROUGH USE OF THE FX(1,---) COMMAND.

LOADERS

AS THE PS/8 SYSTEM DOES NOT HAVE LOADERS IN CORE AT ALL TIMES, ONE EXISTS IN FIELD 1 (STARTING ADDRESS 3777) TO FACILITATE OVERLAY READIN- NOTE THAT TEXT AND VARIABLES WILL OVERWRITE THIS WHEN NECESSARY AS FOCL/S IS USED.

TEXT

IN FIELD 1: STARTS AT 1340 BUILDING TOWARD 7577.

VARIABLES

HAVE BEEN MOVED TO FIELD 1 TO ALLOW (1) ADDITIONS TO THE INTERPRETER IN FIELD Ø AND (2) TRADEOFF BETWEEN VARIABLES AND TEXT. THE FPP, GETVAR, TDUMP, AND ERASE ROUTINES HAVE BEEN MODIFIED TO BEGIN VARIABLE STORAGE AT THE UPPER LIMIT OF FIELD ONE AND BUILD DOWNWARDS. THIS IS TO AVOID ERASING VARIABLES WHENEVER TEXT IS MODIFIED. THE ONLY OBSERVABLE DIFFERENCE BETWEEN THIS AND 8K-FOCAL IS THAT

(1) FOR ALL BUT MAXIMUM SIZED PROGRAMS MORE VARIABLES ARE AVAILABLE.
(2) 'TDUMP' DUMPS VARIABLES IN REVERSE ORDER TO FOCAL/69 I.E. THE MOST RECENTLY CREATED VARIABLE IS DUMPED FIRST.

MODIFICATIONS HAVE BEEN MADE TO GETVAR,SET, AND FOR TO IMPLIMENT ARRAY VARIABLES
ROUTINES MODIFIED OR DELETED

- FPNT
- XRTL6 (MOVED)
- TDUMP
- ERASE
- FOR/SET
- ASK/TYPE
- GETVAR

PUSH DOWN LIST

THE PDL RESIDES IN FIELD 1 WITH THE TEXT AND VARIABLES. AS OPPOSED TO FOCAL/69 THE PDL IS BUILT FROM THE TOP OF TEXT UPWARDS RATHER THAN THE TOP OF CORE DOWNWARDS.
THIS WAS TO ALLOW VARIABLE STORAGE SUSCH THAT MODIFYING TEXT DID NOT ERASE VARIABLES.
ROUTINES MODIFIED OR DELETED

```
INPUTX
START
GONE
XPUSHA
XPOPA
XPUSHF
XPOPF
RETRN
```

ERRORS

THE ERROR RECOVERY ROUTINE HAS BEEN MODIFIED TO GENERATE MORE UNIQUE ERROR MESSAGES. WHAT HAS BEEN DONE IS IF THE ERROR WAS TABLE DRIVEN IT PRINTS THE TABLE POINTER AS A LINE NUMBER. THE ERROR TABLE IS GIVEN LATER.

USER DEFINED FUNCTIONS

THE USER MAY CODE MACHINE LANGUAGE FUNCTIONS FOR USE IN FOCL/S THROUGH THE FX FUNCTION BRANCH WITHOUT SACRIFICING PRESENTLY EXISTING FUNCTIONS OR FUNCTION NAMES. ENTRY OF THE STARTING ADDRESS OF THE FUNCTION INTO THE FXGO TABLE WILL CAUSE BRANCH TO THAT ADDRESS WITH 'CHAR' EQUAL TO CHARACTER FOLLOWING THE N IN FX(N,--)(COMMA IN THIS CASE)

```
FXGO=.          (3773)
OCTANX  /FX(8)
DECNX   /FX(10)
FCOR    /FX(1)
FXCT    /FX(2)
FAND    /FX(3)
FXNO    /FX(4)-NOT IMPLIMENTED
FXNO    /FX(5)
FXNO    /FX(6)
FXNO    /FX(7)
```

SEVERAL ROUTINES EXIST FOR EVALUATING ARGUMENTS. TO MOVE PAST COMMA AND EVALUATE NEXT ARGUMENT:

```
PUSHJ
 NXTARG
XXXX    /NEXTARG DOES NOT EXIST
XXXX    /NORMAL RETURN WITH ARG IN FLAC
```

OTHER USEFUL ROUTINES ARE:
TO SET 'FLAC' EQUAL TO INTEGER VALUE OF ACCUMULATOR:

```
        JMS I PXFIX
PXFIX,  XFIX
```

TO DO THE ABOVE AND DO A NORMAL FUNCTION RETURN:

```
        JMP I PSETFLAC
PSETFL, SETFLAC
```

HOLE AVAILABLE TO USER IS:4454-5377

ARRAY STORAGE

THE LIBRARY LIMIT COMMAND DEFINES THE BOUNDARY BETWEEN RESERVED ARRAYS AND THE TEXT/PDL/VARIABLE AREA OF FIELD 1. WHEN THIS COMMAND IS EXECUTED ALL PRESENT VARIABLES ARE ERASED. IF AN ATTEMPT IS MADE TO SET THIS LIMIT LOWER THAN THE UPPER BOUND OF TEXT CURRENTLY IN CORE, THEN THE COMMAND IS IGNORED AND AN ERROR MESSAGE IS PRINTED.

XOD COMPATIBILITY

THE INTERRUPT PROCESSOR BRANCH HAS BEEN CHANGED TO ALLOW DIRECT USE OF 'XOD' AS A DEBUGGING PACKAGE FOR FOCL/S. A BINARY TAPE IS AVAILABLE, WITH PATCHES TO MAKE THE COMMAND STRUCTURE SIMILAR TO ODT, OF XOD THAT IS DESIGNED TO RUN WITH FOCL/S. THE BEAUTY OF XOD IS THAT IT RUNS WITH THE INTERRUPT ON AND RESTORS KEYBOARD FLAGS TO THE STATE IT EXISTED WHEN BREAKPOINT WAS REACHED.

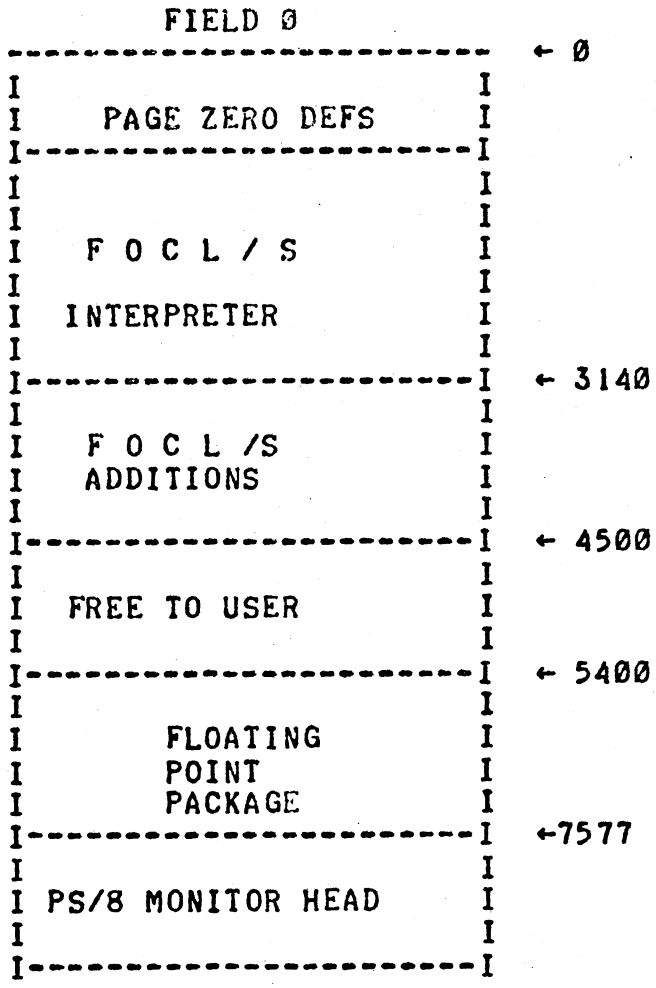
HINTS AND KINKS

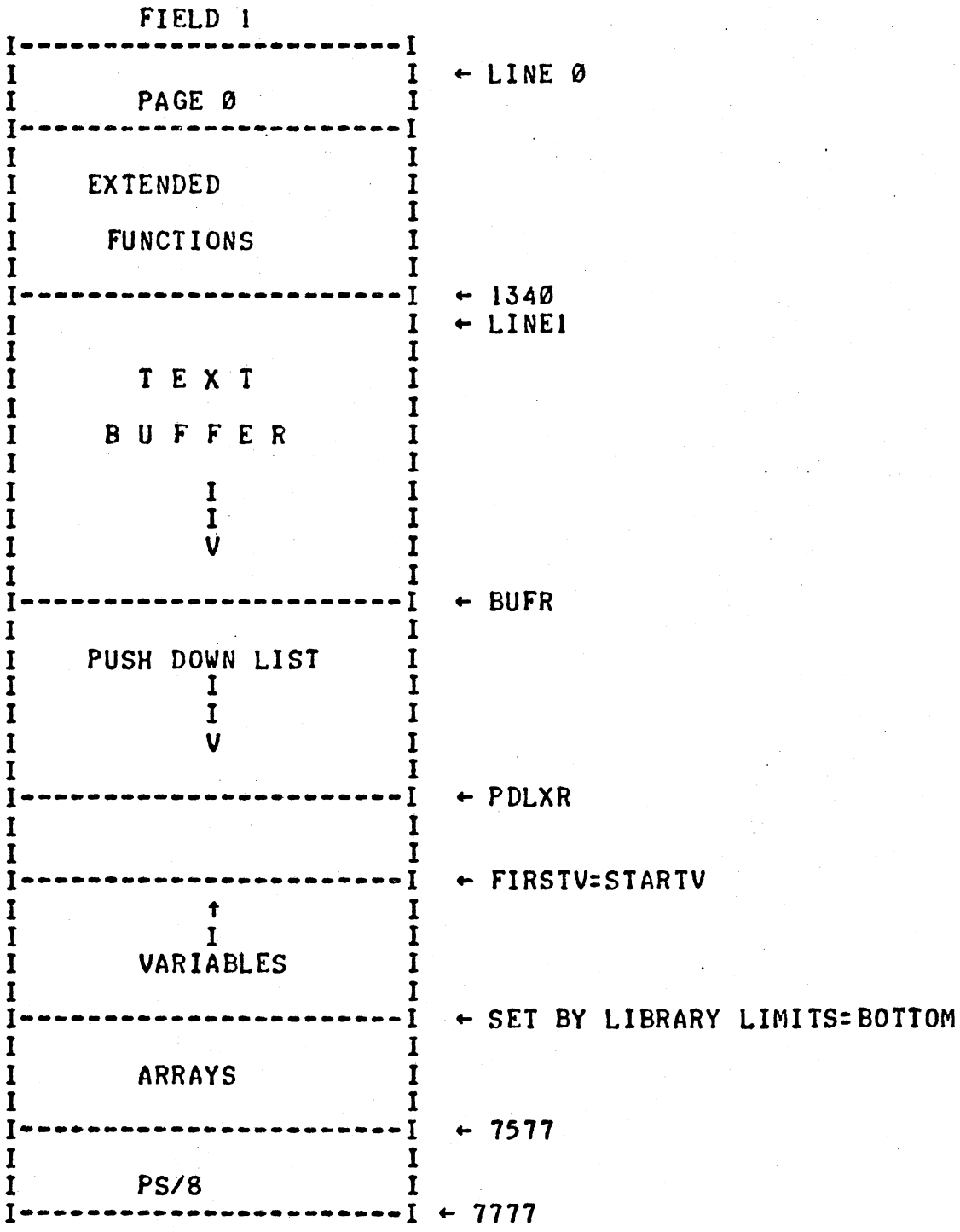
-TO TAKE OUT EQUALS: FX(1,3165,160)
- TO TAKE OUT COLON: FX(1,658,3712)
- TO CHANGE CNT/C TO CNT/O: FX(1,1410,3953)
FX(1,3923,143)
- TO CHANGE '←' TO CNT/U: FX(1,3924,149) - [COMMAND LINES]
FX(1,3583,149) - [FPP. INPUT]
- TO IMPLIMENT LINE NUMBER COMPUTATION FROM ARITHMETIC EXPRESSIONS AT ALL TIMES (I.E. DO 2*X IS LEGAL):

```
S Z=FX(1,196,3065)
```

-TO TURN OFF TELETYPE ECCO: FX(1,1140,3712)
-TO RESTORE TELETYPE ECCO: FX(1,1140,2404)

CORE MAP


FIELD 0
← 0
PAGE ZERO DEFS
F O C L / S
INTERPRETER
← 3140
F O C L /S
ADDITIONS
← 4500
FREE TO USER
← 5400
FLOATING
POINT
PACKAGE
←7577
PS/8 MONITOR HEAD


FIELD 1
← LINE Ø
PAGE Ø
EXTENDED
FUNCTIONS
← 134Ø
← LINE1
T E X T
B U F F E R
← BUFR
PUSH DOWN LIST
← PDLXR
← FIRSTV=STARTV
VARIABLES
← SET BY LIBRARY LIMITS=BOTTOM
ARRAYS
← 7577
PS/8
← 7777

ERROR DIAGNOSTICS - FOCL/S

MESSAGE	ORIGIN	MEANING
?00.00	200*	INITIAL STARTUP MESSAGE.
?01.00	177*	USER TYPED 'CTRL/C'.
?01.40	250*	ILLEGAL STEP OR LINE-NUMBER USED.
?01.78	316*	GROUP NUMBER TOO LARGE.
?01.96	340*	DOUBLE DECIMAL POINTS IN LINE NUMBER
?01.:5	351*	LINE-NUMBER TOO LARGE.
?01.;4	362*	GROUP ZERO IS ILLEGAL AS LINE-NUMBER.
?02.33	441	NONEXISTANT GROUP REFERENCED BY "DO".
?02.53	465	EXCESS RIGHT TERMINATORS ENCOUNTERED.
?02.72	510	IMAGINARY SQUARE-ROOT REQUESTED.
?02.80	520	PROGRAM TOO LARGE (PUSHDOWN-LIST OVERFLOW).
?03.08	610	NONEXISTANT LINE AFTER "GOTO" OR "IF"
?03.31	637	ILLEGAL COMMAND USED.
?04.49	1057	LEFT OF "=" IN ERROR IN "FOR" OR "SET".
?04.64	1077	EXCESS RIGHT TERMINATORS.
?04.71	1107	ILLEGAL "FOR" COMMAND TERMINATOR.
?04.74	1112	LOG OF Ø REQUESTED
?05.53	1265	BAD ARGUMENT IN "MODIFY".
?06.00	1401	";" ILLEGAL AS ARGUMENT TERMINATOR.
?06.01	1402	FOR LOOP TERMINATES BADLY.
?06.08	1410	ILLEGAL USE OF FUNCTION OR NUMBER.
?06.55	1467	STORAGE FILLED BY VARIABLES
?07.22	1626*	OPERATOR MISSING OR DOUBLE "E" IN NUMBER.
?07.38	1646*	NO OPERATOR PRECEEDS PARENTHESES.
?07.:9	1755*	NO ARGUMENT TO THE FUNCTION.
?07.;6	1764*	ILLEGAL FUNCTION NAME OR DOUBLE OPERATORS
?08.48	2060	PARENTHESES DO NOT MATCH.
?08.<1	2172	FRAN FUNCTION NOT AVAILABLE.
?08.<2	2173	FADC FUNCTION NOT AVAILABLE.
?09.01	2202	FNEW FUNCTION NOT AVAILABLE.
?09.02	2203	FCOM FUNCTION NOT AVAILABLE.
?09.12	2214	BAD ARGUMENT IN "ERASE" COMMAND.
?10.93	2535	STORAGE FILLED BY TEXT.
?11.35	2643*	INPUT TOO FAST FOR OUTPUT.

?13.97	3331	SPECIAL VARIABLE TO LEFT OF "=" IN FOR STATEMENT
?13.94	3336	ILLEGAL LIBRARY CALL
?14.48	3460	ARGUMENT MISSING IN FX
?14.82	3522	CONVERSION OF NON-OCTAL # REQUESTED
?15.10	3612	NO ARGUMENTS IN FX(3,--).
?15.15	3617	NO SECOND ARGUMENT IN FX(3,--).
?15.38	3646	ARG1 WRONG IN FX(ARG1,--)
?15.47	3657	VARIABLE IS NOT RESERVED - HENCE "CLOSE" ILLEGAL
?15.60	3674	ARRAY NAME ALREADY IN USE
?15.65	3701	ALL ARRAY NAMES IN USE
?15.92	3734	ONLY 4 SUBSCRIPTS ALLOWED
?16.46	4056	TO MAY SUBSCRIPTS FOR THIS VARIABLE
?16.62	4076	WRONG NUMBER OF SUBSCRIPTS
?16.73	4111	OUTSIDE COMMON ARRAY AREA; < LOWER LIMIT
?16.77	4115	OUTSIDE ARRAY AREA; > UPPER LIMIT
?16.:4	4150	ARRAY 'BASE' OVERLAPPING TEXT (L L,BASE - TO LOW)
?17.51	4263	NO ARGUMENT IN FX(1,--)
?17.71	4307	NO ARGUMENTS IN FX(2,--).
?23.36	5644	INPUT TO FAST FOR FPP.
?26.73	6543	EXPONENT NEGATIVE OR TOO LARGE
?28.73	7111	DIVISION BY ZERO REQUESTED.
?30.40	7450	ARGUMENT MISSING
?30.:2	7546	ARGUMENT MISSING IN FDYS.