FOCAL-148B (8K)

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

D. E. Wrege

Submitted by: John Alderman
Georgia Tech
Atlanta, Georgia

November 28, 1970

PAL-8

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. 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 material, and no responsibility is assumed by these parties in connection therewith.
FOCL/S, AN EXPANDED LANGUAGE FOR SMALL COMPUTERS, BASED ON FOCAL

DESCRIPTION

FOCL/S IS A ONE USER VERSION OF FOCAL/69 FOR 8K PDP-8/12 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 0 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
- $ 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 FOCIL/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, 0, 2, 2, 2
```

Will open a 4-dimensional array named X of maximum length 2 in each dimension. The variable origin parameter is 0, 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, EXPRESSION
```

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 0 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 FOCAL/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, EXCEPTALLY 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 TIME CONSUMING. HENCE, IN FOR LOOPS ETC, WHERE ONE WISHES TO SAVE TIME, EXPLICIT LINE NUMBER EXPRESSIONS ARE DESIRABLE.

--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+FSQRT(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 RANGE 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 0 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 IMPLEMENT 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 such that modifying text did not erase variables.

Routines modified or deleted

INPUTX
START
GONE
XPUSH
XPOPA
XPUSHF
XPOPF
RETN

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 FOCAL/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 implemented
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 /NXTARG 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 restores
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 implement 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

---------------
<table>
<thead>
<tr>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE ZERO DEFS</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>F O C L / S</td>
</tr>
<tr>
<td>INTERPRETER</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>F O C L /S</td>
</tr>
<tr>
<td>ADDITIONS</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>FREE TO USER</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>FLOATING</td>
</tr>
<tr>
<td>POINT</td>
</tr>
<tr>
<td>PACKAGE</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>PS/8 MONITOR HEAD</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

\[ \leftarrow 0 \]

\[ \leftarrow 3140 \]

\[ \leftarrow 4500 \]

\[ \leftarrow 5400 \]

\[ \leftarrow 7577 \]
FIELD 1

PAGE 0

EXTENDED FUNCTIONS

TEXT BUFFER

V

PULL DOWN LIST

V

FIRSTV=STARTV

VARIABLES

SET BY LIBRARY LIMITS=BOTTOM

ARRAYS

7577

PS/8

7777
### Error Diagnostics - FOCL/S

<table>
<thead>
<tr>
<th>MESSAGE ORIGIN</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>?00.00 200*</td>
<td>Initial Startup Message.</td>
</tr>
<tr>
<td>?01.00 177*</td>
<td>User Typed 'CTRL/C'.</td>
</tr>
<tr>
<td>?01.40 250*</td>
<td>Illegal Step or Line-Number Used.</td>
</tr>
<tr>
<td>?01.78 316*</td>
<td>Group Number Too Large.</td>
</tr>
<tr>
<td>?01.96 340*</td>
<td>Double Decimal Points in Line Number</td>
</tr>
<tr>
<td>?01.5 351*</td>
<td>Line-Number Too Large.</td>
</tr>
<tr>
<td>?01.4 362*</td>
<td>Group Zero is Illegal as Line-Number.</td>
</tr>
<tr>
<td>?02.33 441</td>
<td>Nonexistent Group Referenced by &quot;DO&quot;.</td>
</tr>
<tr>
<td>?02.53 465</td>
<td>Excess Right Terminators Encountered.</td>
</tr>
<tr>
<td>?02.72 510</td>
<td>Imaginary Square-Root Requested.</td>
</tr>
<tr>
<td>?02.80 520</td>
<td>Program Too Large (Pushdown-List Overflow).</td>
</tr>
<tr>
<td>?03.08 610</td>
<td>Nonexistent Line After &quot;GOTO&quot; or &quot;IF&quot;</td>
</tr>
<tr>
<td>?03.31 637</td>
<td>Illegal Command Used.</td>
</tr>
<tr>
<td>?04.49 1057</td>
<td>Left of &quot;=&quot; in Error in &quot;FOR&quot; or &quot;SET&quot;.</td>
</tr>
<tr>
<td>?04.64 1077</td>
<td>Excess Right Terminators.</td>
</tr>
<tr>
<td>?04.74 1112</td>
<td>Log of 0 Requested</td>
</tr>
<tr>
<td>?05.53 1265</td>
<td>Bad Argument in &quot;MODIFY&quot;.</td>
</tr>
<tr>
<td>?06.00 1401</td>
<td>&quot;;&quot; Illegal as Argument Terminator.</td>
</tr>
<tr>
<td>?06.01 1402</td>
<td>For Loop Terminates Badly.</td>
</tr>
<tr>
<td>?06.08 1410</td>
<td>Illegal Use of Function or Number.</td>
</tr>
<tr>
<td>?06.55 1467</td>
<td>Storage Filled by Variables</td>
</tr>
<tr>
<td>?07.22 1626*</td>
<td>Operator Missing or Double &quot;E&quot; in Number.</td>
</tr>
<tr>
<td>?07.38 1646*</td>
<td>No Operator Precedes Parentheses.</td>
</tr>
<tr>
<td>?07.t 1755*</td>
<td>No Argument to the Function.</td>
</tr>
<tr>
<td>?07.6 1764*</td>
<td>Illegal Function Name or Double Operators</td>
</tr>
<tr>
<td>?08.48 2060</td>
<td>Parentheses Do Not Match.</td>
</tr>
<tr>
<td>?08.* 2172</td>
<td>Fran Function Not Available.</td>
</tr>
<tr>
<td>?08.* 2173</td>
<td>Fadc Function Not Available.</td>
</tr>
<tr>
<td>?09.01 2202</td>
<td>Fnew Function Not Available.</td>
</tr>
<tr>
<td>?09.02 2203</td>
<td>Fcom Function Not Available.</td>
</tr>
<tr>
<td>?10.93 2535</td>
<td>Storage Filled by Text.</td>
</tr>
</tbody>
</table>
115.07 3331 SPECIAL VARIABLE TO LEFT OF "=" IN FOR STATEMENT
115.94 3336 ILLEGAL LIBRARY CALL
114.48 3460 ARGUMENT MISSING IN FX
114.82 3522 CONVERSION OF NON-OCTAL # REQUESTED
114.10 3612 NO ARGUMENTS IN FX(3,-).  
115.15 3617 NO SECOND ARGUMENT IN FX(3,-).
115.38 3646 ARG1 WRONG IN FX(ARG1,-)  
115.47 3657 VARIABLE IS NOT RESERVED - HENCE "CLOSE" ILLEGAL
115.60 3674 ARRAY NAME ALREADY IN USE
115.65 3701 ALL ARRAY NAMES IN USE
115.92 3734 ONLY 4 SUBSCRIPTS ALLOWED
116.46 4056 TOO MANY SUBSCRIPTS FOR THIS VARIABLE
116.62 4076 WRONG NUMBER OF SUBSCRIPTS
116.73 4111 OUTSIDE COMMON ARRAY AREA; < LOWER LIMIT
116.77 4115 OUTSIDE ARRAY AREA; > UPPER LIMIT
116.14 4150 ARRAY 'BASE' OVERLAPPING TEXT (L L,BASE - TO LOW)
117.51 4263 NO ARGUMENT IN FX(1,-)  
117.71 4307 NO ARGUMENTS IN FX(Y,-).
123.36 5644 INPUT TO FAST FOR FPP.
126.73 6543 EXPONENT NEGATIVE OR TOO LARGE
128.78 7111 DIVISION BY ZERO REQUESTED.
150.40 7450 ARGUMENT MISSING
150.72 7546 ARGUMENT MISSING IN FDYS.