KERNFORSCHUNGSANLAGE JÜLICH
GESELLSCHAFT MIT BESCHRÄNKTER HAFTUNG
Zentralinstitut für Angewandte Mathematik

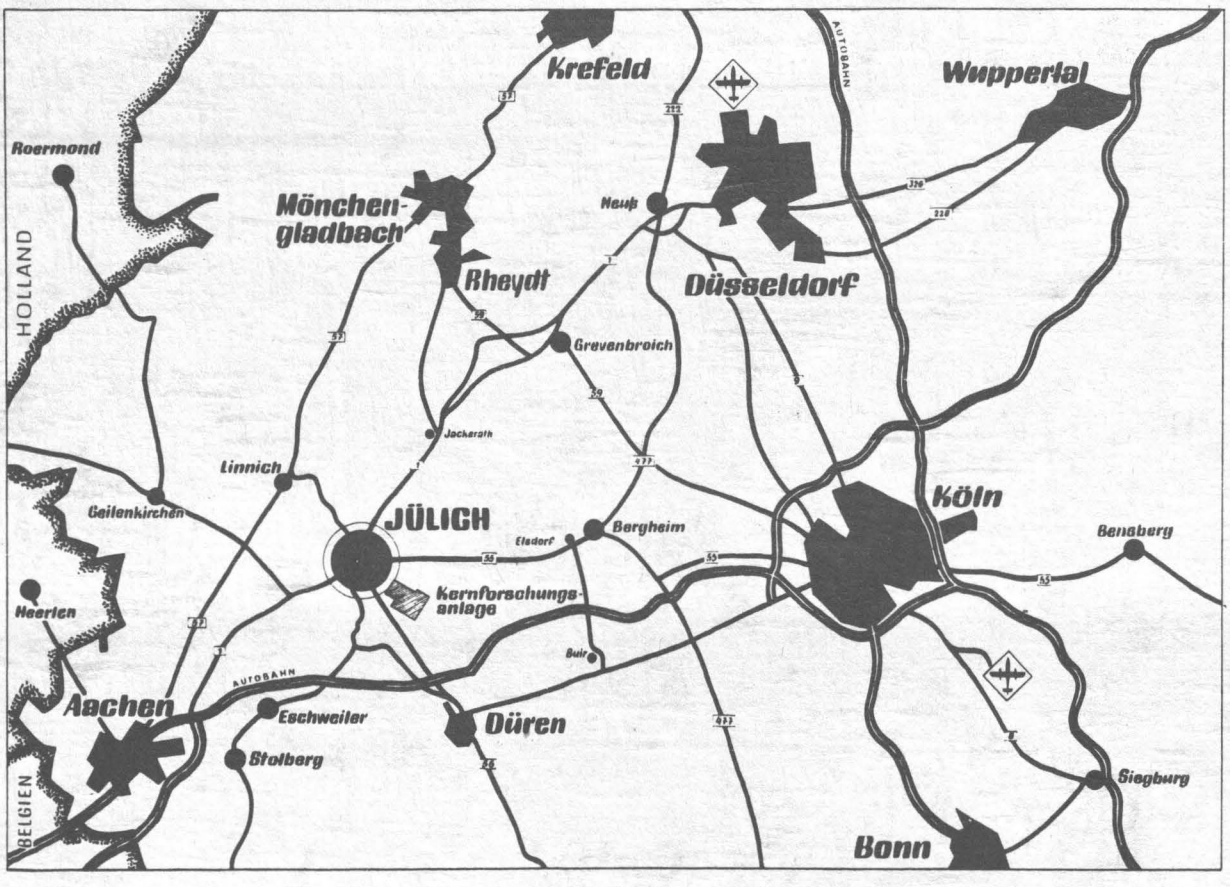
The KFA FORMAC Preprocessor

by

R. Schwerdt

Jül - 676 - MA
Juli 1970

Als Manuskript gedruckt



Berichte der Kernforschungsanlage Jülich – Nr. 676

Zentralinstitut für Angewandte Mathematik Jül – 676 – MA

Dok.: Compiler Construction
Symbolic Mathematical Computations
FORMAC

DK: 681.142.4.002.2
681.142.1.01:681.3.06
681.142.2 „FORMAC"

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH,
Jülich, Bundesrepublik Deutschland

The KFA FORMAC Preprocessor

by

R. Schwerdt

To be presented at SHARE XXXV, Montreal, August 1970

About this manual

The Institute for Applied Mathematics of the KFA Jülich developed a new preprocessor for PL/I FORMAC which has been in use since April 1970 with good success. There are many reasons which forced us to write a quite new precompiler, the most important of which are: some inconvenient properties of the appropriate IBM preprocessor, a planned extension of the language and some FORMAC projects at our installation as the FORMAC INteractive System for TERminals (FINSTER). This paper does not contain a full description of the FORMAC language but should be regarded as a guide for programmers using the KFA preprocessor. Although it applies mainly to Version 0 it remains valid for all subsequent versions.

The task of a FORMAC preprocessor

PL/I FORMAC is a programming language which may be regarded as a superset of PL/I, i. e. all PL/I facilities as program control, loop control, I/O, etc. are available in FORMAC, too. In other words: any legal PL/I statement is also a legal FORMAC statement. In literature you often find the terms "host language" and "guess language" respectively. The task of the preprocessor is to translate mere FORMAC into legal PL/I by scanning for the special FORMAC macros, functions and control statements. Some more details about the processing phase can be found in the chapter "Execution of FORMAC". Normally a FORMAC program passes through four or three steps .

Step 1 : The program is read in via a data set named SYSIN and will be processed. The generated PL/I program (which is hopefully correct) is stored on a temporary data set named SYSUT3, whose characteristics are installation defined.

Step 2 : The generated program will be compiled by the PL/I (F) compiler. The program may contain macro statements (PL/I statements with a leading %); thus additional source programs or parts may be included from SYSLIB.

Step 3 : The Linkage Editor produces an executable load module.

Step 4 : In the last step the program is executed.

Steps 3 and 4 can be merged to a single step if you use a load-and-go program (e.g. the IBM loader) at your installation.

In march 1970 the SHARE Symbolic Mathematical Computation Project compiled a number of resolutions containing a list of 11 FORMAC Preprocessor Design Points, Interface Points, Object Time Housekeeping Points and two Redesign Points for FORMAC Functions. In short the first group refers to known errors of the IBM preprocessor, possible solutions and proposed extensions, last of which are regarded as most important.

The following list is an abstract of the Design Points.

(1) The name of the main procedure should be able to start in column 2.
(2) The preprocessor should be able to handle the PROCESS statement.
(3) The preprocessor should not change any number of blanks in character strings to one.
(4) There should be a DECK/NODECK option to provide punched output from the preprocessor.
(5) There should be a SOURCE/NOSOURCE option to provide for suppression of original unpreprocessed text.
(6) There should be a COMPACT/NOCOMPACT option to permit a readable form of a preprocessed text to be output, i. e. statements specified on separate cards should not be run together.
(7) Output of original text by the preprocessor should be numbered to correspond with the compile phase (debugging aid).
(8) Output of the preprocessor should include columns 73-80 of the original text (rather convenient if you use an RJE-system at your installation).
(9) The FORMAC PL/I functions LOP, ARITH, INTEGER and NARGS should be allowed within double quotes.
(10) The double quote operator should be allowed in all FORMAC expressions.
(11) There should be no restrictions upon the CONVERT statement.

Design Points 6 and 9 are not yet realized in Version Ø of the KFA FORMAC Preprocessor. As for number 6 we think that there is no need for such an option; at present the NOCOMPACT option applies generally; the IBM precompiler runs all statements together.

Execution of FORMAC

Comparing an original FORMAC program with the precompiled version you will probably note that the macros, functions and some control statements are translated into a fixed number of PL/I-declares, subroutine calls and function references ; mere PL/I statements are left unchanged as well as comments and literal constants. Furthermore some control statements are removed altogether. The keyword "FORMAC_OPTIONS" is replaced by a DECLARE for some object time routines starting with the characters DENFMC; to the right an additional alphameric character is attached. It should be noted, that FORMAC itself is actually executed by these routines which are part of the "FORMAC SYSTEM". These modules are called run-time routines.

The characteristic correspondence of the FORMAC macros and functions to their individual routine is listed below:

LET	DENFMC1
PRINT_OUT	DENFMC2
FIXEDA	DENFMCE
FLOATA	DENFMCD
ATOMIZE	DENFMC8
SAVE	DENFMC4
CHAREX	DENFMCH
FORM	DENFMC1
RET	DENFMC1
CET	DENFMC1
OPTSET	DENFMC3
IDENT	DENFMCG
INTEGER	DENFMCA
ARITH	DENFMC9
NARGS	DENFMCC
LOP	DENFMCB
EQUIV	DENFMCG

Besides the keywords listed above there are still some FORMAC statements which are not expanded by the preprocessor, instead they are control statements only affecting the processor's translation. At present there are five

of them:

```
CONVERT...
OPTSET(PRINT)      OPTSET(NOPRINT)
OPTSET(SAVE)       OPTSET(NOSAVE)
```

Two examples for a better understanding:

(a)
```
I=2;
LET(   DV = DERIV(  F, X, "I"  )  );
```

is translated to

```
I=2;
CALL DENFMC1('DV=DERIV(F,X,('||I||'))');
```

(b)
```
CONVERT FIXED(J);
OPTSET(SAVE);
  .
  .
LET( A("J") = A(J+1)-X*A(J); "J"=J+2 );
```

is translated to

```
 DO;
CALL DENFMCE('J=31622',(J));
CALL DENFMC4('A(J)=A(J+1)-X*A(J)');
CALL DENFMC4('J=J+2');
J=DENFMCA('J');
 END;
```

Note that blanks and double quotes within the macros are removed in the preprocessed text. The control statement OPTSET(SAVE) causes all LET's to be handled as SAVE's. Like OPTSET(PRINT) it has no universal scope.

The processing phase

The following section gives you an abstract of the advantages and properties of the KFA FORMAC preprocessor. In addition some practical hints for FORMAC users are given.

Input to FORMAC preprocessor

A FORMAC program is input to the precompiler via a data set with the DD name SYSIN. The margin for the source statements is column 2 and 72 respectively. The input lines are listed with a unique number to the left when the SOURCE option is on. The keyword "FORMAC_OPTIONS" must be specified physically before the appearance of FORMAC; otherwise the program is left unchanged. In other words: all FORMAC keywords preceding "FORMAC_OPTIONS" are not identified. In respect to the IBM preprocessor this is a quite new interpretation of "FORMAC_OPTIONS". Furthermore you should note that this rule applies to every external program in the input stream. Error messages, if any, are interspersed within the text; diagnostics cannot be suppressed.

Back Reference

The generated program is marked in the last eight columns with the number of the corresponding line of the FORMAC program. This is the same technique as in the PL/I macro facility. Its purpose is to help you debugging your program, because error messages printed out during execution time refer to the statement numbers of the preprocessed program.

Using the PROCESS statement

The PROCESS statement is accepted by the KFA FORMAC preprocessor. Thus several external programs can be processed in a batch compilation. Note that the * must be punched in column 1, otherwise the line is treated as part of the current program; furthermore the trailing semicolon must be coded. Having found a legal PROCESS statement the preprocessor performs some checks, forces a new page on the listing, and starts a new compilation, i. e. the

same rules as in "Input to the FORMAC Prepr." apply.

Edit of the preprocessed program

Besides the back reference one of the most important rules for the program is to leave the source deck as unchanged as possible. Above all mere PL/I statements, literal constants and comments are affected by this rule. If you specify an item in column k, you can find it in column k+d, where d is a proper displacement depending on the grade of substitution in the line involved. Interspersed blanks are regarded as significant ones. However, these rules do not apply to macros like LET, SAVE, etc., whose expansion is started on a new line.

Example:
```
J=INTEGER(K) *2;    A=B(J);
will be expanded to
J=DENFMCA('K') *2;    A=B(J);
```

The preprocessor generates the statements to 80 byte card images within the columns 2 and 80; therefore an appropriate source margin must be specified for step 2 (compilation by the PL/I F-compiler). The expansion of macros with two or more substatements or the insertion of proper conversions causes the precompiler to generate a DO group containing all statements. This is necessary because you may specify

```
IF B THEN SAVE(A;B;C);
     ELSE ...
```

At present the PL/I (F) compiler generates a single MVI instruction for the DO; as well as for the END;. But this may be eliminated by specifying the NOSTMT option for the PL/I compiler.
Keywords, strings and other logical units may be separated over two lines - the preprocessor accepts it; but generally the truncation will not be copied to the generated program.

Options at precompile time

Two options may be specified at precompile time:

SOURCE / NOSOURCE or S / NS
DECK / NODECK or D / ND

The underscored alternatives are default and need not be specified. The options are passed using the PARM-field of the EXEC card. The abbreviations should be preferred.

SOURCE	the original program is listed on SYSPRINT; every external program is started on a new page.
NOSOURCE	the print-out of the FORMAC program is suppressed; the print of the diagnostics is not affected.
DECK	the generated program is punched on SYSPUNCH in 80 byte card images; note that the back reference is not punched.
NODECK	no deck supplied.

If contradictory options are specified the result is undefined.

New rules for the CONVERT statement

The KFA preprocessor accepts more than one CONVERT statement. The keywords FIXED and/or FLOAT with an attached list of variables may appear in any order and more than once in a single CONVERT. Note that the total number of all variables specified is restricted to an implementation defined value (default: 99), because the preprocessor has to construct an internal table for the names. The only restriction upon the CONVERT statement is that it must precede all FORMAC macros except FORMAC_OPTIONS and OPTSET. In respect to the IBM precompiler this rule is a convenient extension. A misplaced CONVERT will be deleted.The CONVERT does not affect the translation of FORMAC functions, although double quotes are permitted for them.

The double quote operator

The double quote is allowed for all macros and functions. But it should not be used indiscriminately. Instead the macros FIXEDA and/or FLOATA should be specified. A better way is the use of CONVERT in combination with the double quote operator (compare the chapter "FIXEDA, FLOATA, INTEGER, ARITH - The CONVERT statement").

New diagnostics

A list of the new error messages which may appear in step one is given in this paper. It should be emphazised that it applies to Version 0, and it is not yet definitive. Some of the messages start with a colon indicating that the detected error is the consequence of the one diagnosed before. If some invalid or unexpected text is deleted by the preprocessor the user is informed; up to eight significant characters are printed out. Furthermore you should note, that the listed actions (e. g. insertion of parentheses or semicolons, deletion of text, etc.) may be incorrect; therefore the correction of errors by the precompiler is declared as an error in the sense of PL/I: you should correct it. In short: Don't make the preprocessor correct your program.

Restrictions

At present two restrictions apply. The maximum length of a FORMAC substatement (significant characters only) and the total length of a single literal constant should not exceed 500 bytes or approximately 6 input lines. If this restriction is violated an abnormal end with a fixed user code (default:2424) will occur. Furthermore the slash and the asterisk of a comment delimiter must be written on a single line.

Combination of options

The FORMAC language permits the combination of executable and non-executable options in a single OPTSET macro. The IBM preprocessor does not support this feature. The KFA precompiler accepts it! But sometimes it is convenient for the user to put a control option like "PRINT" or "SAVE" in a single OPTSET

on a separate line, so that it can be easily removed.

Implemented Control Statements, Macros and Functions

```
FORMAC_OPTIONS [(PRINT<SAVE>)]
CONVERT ...
OPTSET(PRINT)    OPTSET(NOPRINT)
OPTSET(SAVE)     OPTSET(NOSAVE)

LET
PRINT_OUT
FIXEDA
FLOATA
ATOMIZE
SAVE
CHAREX
FORM
RET
CET
OPTSET

IDENT
INTEGER
ARITH
NARGS
LOP
EQUIV
```

Some of the features listed above are not described in the manual "PL/I-FORMAC Interpreter", but they are accepted by the IBM preprocessor and therefore they had been implemented in the KFA's, too. It is very easy to derive their meaning from the expansion shown in step 2.

FORM, RET, and CET

FORM, RET and CET are simply modifications of the wellknown LET. They enable you to introduce legal FORMAC statements and/or names into the FORMAC system during execution time via a PL/I character string variable. Thus you have the possibility to put unedited expressions in the input stream.

The general format is:

```
FORM(pstring1;pstring2 ;...);
RET(pstring1 [=fname1] ;pstring2 [=fname2] ;...);
CET(fname1[=pstring1] ;fname2 [=pstring2];...);
```

"pstring" indicates a PL/I character string variable,"fname" a FORMAC name.

General rules:

(1) Literal constants are allowed for "pstring".

(2) If the name of the PL/I variable is equal to the selected FORMAC name, the second item of the sub-statement in RET or CET including the = sign may be omitted.

Some examples will show you the advantages of them:

(1)
```
DECLARE S1 CHARACTER(100);
     S1 = 'X=(A+B)**2' ;
     FORM(S1);
```
is equivalent to
```
   LET( X=(A+B)**2 ) ;
```
or
```
   LET( "S1" );
```
but the last one is not recommended.

(2)
```
DECLARE S1 CHARACTER(100) ;
    .
    .
LOOP:GET FILE(DATA_IN) EDIT( S1 ) (A(100));
     FORM( S1 );
     GO TO LOOP;
GO_ON:...
```

(3) a]
```
DECLARE S2 CHARACTER(100);
```
b]
```
LET( S2=(A+B)**2 );
    .
    .
```

```
c]  S2 = 'NEWNAME' ;
d]  RET( S2 ); /* OR: RET(S2=S2); */
is equivalent to
    LET( S2=(A+B)**2 );
    LET( NEWNAME=S2 );
Note: In line b the variable S2 is a FORMAC variable;
      in lines a, c and d it is a PL/I variable!
```

```
(4) DECLARE B CHARACTER(100);
    B = 'DERIV(F,X,N)' ;
    CET( S=B );
is equivalent to
    LET( S=DERIV(F,X,N) );
```

The macro FORM is the most important one and may be seen as the counterpart to CHAREX.

EQUIV

The function EQUIV is a modified version of the function IDENT with the same attributes and syntax:

$$\text{EQUIV}(a_1;a_2) \ .$$

The only difference is that the FORMAC level function EXPAND is applied to both arguments before the comparison is performed. Thus we may say

$$\text{EQUIV}(a_1;a_2) := \text{IDENT}(\ \text{EXPAND}(a_1);\text{EXPAND}(a_2)\ )$$

List of the implemented options

The following options may be specified in an OPTSET statement.

TRANS	NOTRANS
INT	NOINT
EXPND	NOEXPND
EDIT	NOEDIT
PROPER	IMPROPER
ERSNP	NOERSNP
CHAR	NOCHAR
CHANGE	NOCHANGE
FLOW	NOFLOW
-	NODUMPS
LINELENGTH	

and the "unexecutable"

PRINT	NOPRINT
SAVE	NOSAVE

Although the options [NO]ERSNP, [NO]CHAR, [NO]CHANGE, [NO]FLOW and NODUMPS are valid, no detailed information is supplied by IBM. The option ERSNP (Error snap) produces a special FORMAC dump when an error occurs. The control option SAVE affects the translation of the program:all LET's, which are found physically behind the setting, are treated as SAVE's until the effect is eliminated by one of the options NOSAVE, PRINT or NOPRINT.
The meaning of the remaining options could not be derived from deliberate tests precisely. We assume that the appropriate facilities are not yet implemented in the FORMAC system.

FIXEDA, FLOATA, INTEGER, ARITH

The CONVERT statement

FORMAC provides a number of facilities for the interface of PL/I and the FORMAC system; you can pass mere PL/I values and make PL/I benefit of the special FORMAC's capabilities. The most simple way to introduce a PL/I value into the FORMAC system is the double quote operator. But you should not use it indiscriminately. The rather trivial statement LET(I="I"); is translated to CALL DENFMC1('I= '||I||')'); . The generated concatenations are time consuming; furthermore the PL/I compiler produces code for the conversion of I to a character string if it is not. A better way for the PL/I-FORMAC Interface are the macros FIXEDA and FLOATA. General format:

```
FIXEDA(fname1=pname1;fname2=pname2;. . .);
FLOATA(fname1=pname1;fname2=pname2;. . .);
```

The numeric variable pname is assigned to fname as a fixed or float variable respectively.
Example b below is preferable to a:

```
a)    DO I=1 TO N;
        LET(  I="I";
              A(I,I)=1);
      END;

b)    DO I=1 TO N;
        FIXEDA( I=I );
        LET( A(I,I)=1 );
      END;
```

ARITH and INTEGER are supported for FORMAC- PL/I Interface. They are used as ordinary functions in the well known way. They return a float or fixed number respectively. If the argument does not evaluate to a numeric value an error message is generated during execution time.

The purpose of the unexecutable CONVERT statement is to facilitate PL/I-FORMAC Interface as well as FORMAC - PL/I Interface. In combination with

the double quotes it makes the preprocessor to insert automatically proper FIXEDA and FLOATA macros and/or INTEGER and ARITH conversions. General format:

$$\text{CONVERT}\quad \text{FIXED}(x_1;x_2;\ldots;x_n)\quad \text{FLOAT}(y_1;y_2;\ldots;y_m)\ ;$$

Note that the keywords FIXED and FLOAT may appear in any order and more than once in a single CONVERT statement. More than one CONVERT is permitted. Some examples show you the effect; a detailed description is given in IBM's "PL/I-FORMAC Interpreter". On the left hand side some typical statements are shown. On the right hand side the expansion (done by the preprocessor) can be seen.

```
CONVERT FIXED(I;J) FLOAT(A);
                                        DO;
                                       FIXEDA(I=I;J=J);
LET(D("I")=D(I)**"J");                 LET(D(I)=D(I)**J);
                                        END;
                                        DO;
SAVE( "J"=FAC(10) );                   SAVE(J=FAC(10));
                                       J=INTEGER(J);
                                        END;
                                        DO;
                                       FIXEDA(I=I);
LET("A"=B(J,"I")-SUM);                 LET(A=B(J,I)-SUM);
                                       A=ARITH(A);
                                        END;
                                        DO;
                                       FIXEDA(J=J);
LET( "J"="J"+1 );                      LET(J=J+1);
                                       J=INTEGER(J);
                                        END;
```

The last example on the left hand side is a somewhat unusual way to increment a PL/I variable J as well as a FORMAC J by 1, but it is a legal statement.

Special consideration for macros

All FORMAC macros allow the specification of several substatements, e.g.

```
LET( A="A"; B="B"; X1=(2*X-A-B)/(B-A));
```

which are separated by a semicolon. But the macros themselves have a trailing semicolon, too, which indicates the end of the macro. Therefore it is not possible to determine the exact end of a FORMAC macro when one or more parentheses in a substatement are missing.
In

```
LET( A=B(J*(K+1));
     DO IS=1 TO NS;
```

the error is quite obvious for the preprocessor as well as for the reader; the second statement of the example below, however, may be regarded as a substatement of the LET:

```
LET( A=B(J*(K+1));
    AP=ARITH(A);
    PUT DATA(AP);
```

The last example shows you rather evidently the difficulties arising in the case of unmatched parens. The consequence is that the KFA preprocessor performs some simple checks mainly based on two rules:

(1) 1.1 The level of parens must be zero at the end of a substatement (left paren: add 1, right paren: subtract 1).
1.2 If it is greater than zero one or more right parens are inserted; furthermore the preprocessor assumes that the end of the macro has been reached.
1.3 If it is less than zero one or more right parens are deleted.

(2) Every alphameric, alphabetic or numeric item within a macro or function must be enclosed in valid special characters - otherwise an error message is issued and processing of the macro involved is terminated.

Some instructive examples are given in the list of the error messages.

Planned extensions

The Institute for Applied Mathematics of the KFA Jülich is going to develop a subroutine technique which would facilitate the establishment and use of a FORMAC program library. The problem is that FORMAC variable names are of universal scope. If in a program a variable , say X, has been defined it is identical to all X's specified in some other programs. It seems not advisable to introduce restrictive coding conventions. We intend to define a simple extension of the FORMAC language which causes for example the precompiler to modify your program in the proper way. Furthermore a statement will be supported, that permits the definition of local variables.

Adding the KFA Preprocessor to your Operating System

The KFA FORMAC Preprocessor is released as an unloaded version which had been produced by the standard utility program IEHMOVE; therefore the user should restore the PDS only with IEHMOVE. Below is a job performing this function. Your system programmer has to fill in the unit-type and volume serial number to match yoursystem's requirements. The unloaded PDS on the tape is named KFAFORM and may be renamed on the disk by the RENAME option of the IEHMOVE utility; the only member is the preprocessor named MINIMAC. Of course, you can catalog the data set using IEHPROGM. In this case you can use the description on page 111 of "PL/I-FORMAC Interpreter", corrected version September 1969, form 360D-03.3.004.

```
//.....  JOB  ....
//L      EXEC  PGM=IEHMOVE
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  scratch data set
//IN       DD  DSN=KFAFORM,UNIT=device,VOL=SER=sr,
//  DISP=(OLD,KEEP),DCB=DEN=density
//OUT      DD  UNIT=device,VOL=SER=sr,DISP=OLD
//SYSIN    DD
   COPY PDS=KFAFORM,TO=device=vol-sr,FROMDD=IN,          CONT
                TODD=OUT,RENAME=lib-name
/
```

Note that no SPACE parameter is needed in the OUT DD card. The IEHMOVE will allocate the exact amount required.

Another way to process PL/I-FORMAC

A possible solution to the problem of precompiling a FORMAC program is the PL/I compile time facility. Mr. R. A. Stone, Western Electric, New Jersey, U.S.A., wrote a set of compile time procedures, which are 'called' from SYSLIB by a %INCLUDE statement. The individual functions, macros and control statements of the FORMAC language are declared as compile time procedures, which expand the appropriate FORMAC statements. The advantages of such a technique are quite obvious:

(a) Generally better diagnostics are supported.
(b) A back reference is supported, i. e. the generated statements are marked in the last columns with a number refering to the line from which they had been generated; this is a fairly good debugging aid.

(c) More than one external program can be compiled in a batch compilation using the PROCESS statement.

(d) The program is not compressed, i. e. end-of-cards are retained.

(e) A DECK option is provided (MD for the PL/I F-compiler)

On the other hand some inconvenient disadvantages arise:

(a) The double quotes cannot be used because it is not a legal character of the 60 and 48 character set respectively. It must be substituted by a legal one; Mr. Stone selected the question mark. Introducing two different characters, e. g. < and >, which facilitate the nesting of the operator, seems preferable but difficult.

(b) The FORMAC language cannot be supported fully. The PL/I macro facility does not allow procedure definitions with a variable number of arguments ; thus only one of the legal alternatives FORMAC_OPTIONS or FORMAC_OPTIONS(PRINT) is permitted. Similar problems arise in the case of the CONVERT statement.

(c) Compile time increases!

```
//         PROC  PL1OPT=,OPTION=
//C        EXEC  PGM=MINIMAC
//STEPLIB  DD    UNIT=2314,VOL=SER=LIB003,DSN=KFAFRMC,DISP=SHR
//SYSPRINT DD    SYSOUT=A                                              *
//SYSUT3   DD    DSNAME=&FRMC,DISP=(,PASS,DELETE),UNIT=2314,
//              SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=88,BLKSIZE=3020)
//CP       EXEC  PGM=IEMAA,PARM='SM=(2,80),&PL1OPT',COND=(12,EQ,C)
//SYSPRINT DD    SYSOUT=A
//SYSLIN   DD    DSN=&OBJSET,DISP=(,PASS),SPACE=(6400,(120)),UNIT=2314,*
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT1   DD    DSN=&SYSUT1,SPACE=(6400,(90)),UNIT=2314
//SYSUT3   DD    DSN=&SYSUT2,SPACE=(6400,(120)),UNIT=2314
//SYSIN    DD    DSNAME=&FRMC,DISP=(OLD,DELETE)
//L        EXEC  PGM=IEWL,PARM='LIST,&OPTION',COND=(5,LT,CP)
//SYSPRINT DD    SYSOUT=A
//SYSLIN   DD    DSN=&OBJSET,DISP=(OLD,DELETE)
//         DD    DDNAME=SYSIN
//SYSLIB   DD    DSN=SYS1.PL1LIB,DISP=SHR
//         DD    DSN=SYS2.DENLIB,DISP=SHR
//SYSUT1   DD    DSN=&SYSUT1,SPACE=(6400,(90)),UNIT=2314
//SYSLMOD  DD    DSNAME=&GO(MAIN),DISP=(NEW,PASS,DELETE),UNIT=2314,    *
//              SPACE=(CYL,(10,1,1))
//G        EXEC  PGM=*.L.SYSLMOD,COND=((5,LT,CP),(5,LT,L))
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSN=&SYSUT1,SPACE=(6400,(90)),UNIT=2314,              *
//              DCB=(RECFM=F,BLKSIZE=900)
```

Above is a listing of a catalogued procedure for precompiling, compiling, link-editing, and executing a FORMAC job. But it should be used as a pattern only, because some of the parameters must be modified to match your installation's requirement. The PROC statement declares two parameters which facilitate the use of the procedure. "PL1OPT" may be used to pass options to the PL/I compiler in step two; the required source margin specification is already defined and should not be specified for a second time. If you use the PROCESS statement in the input stream the appropriate source margin must be specified in it; version 0 of the preprocessor does not yet insert it automatically. "OPTION" is dedicated for the Linkage Editor options. Some more hints for a FORMAC procedure are given on page 112 of "PL/I-FORMAC Interpreter".

Examples

On the following pages some instructive examples are listed. The best way to learn the advantages and properties of the KFA preprocessor is to use it! The first procedure evaluates the Bernoulli numbers in rational form, i. e. exactly; the output is not shown. In the second example the expression $\sin^n(x)$ for a positive n is transformed into a trigonometric polynomial in sines or cosines of multiples of x. Both examples were taken from a number of exercises of Mr. D. Valenzuela. The only purpose of the last example is to show you the diagnostic feature of the precompiler. The reader who is familiar to FORMAC should analyze it for a better understanding and to get an insight into the new way of processing a program. Note that the format of the listings shown is not the original one because it did not match the size of the booklet.

Error messages

The list contains a short explanation, the preprocessor's action, what the user should do, and a classification of the error:

W - Warning
E - Error
S - Severe Error

UNMATCHED COMMENT DELIMITER OR QUOTES. LOOK AT LINE xxxxx S

Prepr. action : None
User action : Correct the error; the comment or literal constant involved starts in line xxxxx. Execution will fail.

UNEXPECTED KEYWORD E

Explanation : A keyword has been specified twice or more, e. g. the keyword 'FORMAC-OPTIONS'.
Prepr. action : The keyword and all text up to the next semicolon outside parentheses is deleted.
User action : Remove the keyword.

RIGHT PAREN IGNORED E

Explanation : A redundant paren has been deleted, e. g. in
LET(A=B(J)));
the last one is ignored.
User action : Check the statement and correct it.

RIGHT PAREN(S) INSERTED E

Explanation : More than one right paren missing, e. g. the statement
LET(A=B(J ; is treated as
LET(A=B(J)) ;
Prepr. action : Insertion of right parens at the end of the statement.
User action : Correct the error or execution may fail.

INVALID DELIMITER.';' ASSUMED E

Explanation : An unexpected comma has been found, e. g. in
LET(A=B,C=DERIV(SIN(X),X)),
two commas are changed to a ';', namely the first and the last one.
Prepr. action : The comma is changed to a semicolon.
User action : Correct all relevant commas.
Otherwise execution may fail.

UNMATCHED " S

Prepr. action : None
User action : Correct the error.
Execution will fail, when the variable(s) involved have not been specified in a CONVERT statement.

INVALID: 'text'
:PAREN MAY BE MISSING E

Explanation : The preprocessor found an illegal combination of syntactical units in a FORMAC-macro; possible error: missing right paren.
E. g.: LET(A=B(J);DO I=1 TO N;
Prepr. action : The rest of the text within the macro following 'text' is regarded as PL/I.
User action : Check the statements.

INVALID: 'text'
:PAREN MAY BE MISSING E
:EXPANSION NOT DONE

Explanation : The preprocessor found an illegal combination of syntactical units in a FORMAC-PL/I-Interface-Function; possible error: missing right paren.
Prepr. action : The rest of the text is regarded as PL/I; instead of appropriate code for the function the comment / - ERROR- / is generated.
User action : Check the statement.

MISPLACED CONVERT STATEMENT E

Explanation : The CONVERT statement must occur physically before the appearance of the first FORMAC macro with the exception of OPTSET.
Prepr. action : The entire CONVERT statement is deleted.
User action : Remove the statement or put it at the beginning of the program.

INVALID:text E

Explanation : Some unexpected text has been found, e. g. in OPTSET(TRANS;!EDIT);
the '!' is invalid.
Prepr. action : text is ignored.
User action : Remove the invalid item.

TOO MANY FIXED VAR. S
TOO MANY FLOAT VAR. S

Explanation : There is an implementation restriction upon the max. number of fixed/float variables specified in all CONVERT statements.
Prepr. action : The rest of the list(s) is ignored.
User action : Cut the list or segment the program using the PROCESS;.

ILLEGAL VAR. IN CONVERT STMT E

Explanation : A subscripted variable has been specified in CONVERT...

Prepr. action : The subscripts are ignored.
User action : Correct the error.

text TOO LONG S

Explanation : A variable in a CONVERT statement is longer than 8 characters.
Prepr. action : The variable is deleted.
User action : Cut the variable or remove it from the list.

DOUBLE DECL. OF text IN CONVERT STMT E

Explanation : The variable text has been specified more than once in a CONVERT statement.
Prepr. action : The double declaration is ignored.
User action : Remove the variable text from the list.

ERROR IN OPTSET-STATEMENT S

Explanation : One or more parens had been found in an OPTSET statement.
Prepr. action : It attempts to delete the invalid text but may fail.
User action : Correct it.

ILLEGAL LINELENGTH E

Explanation : The linelength specified is beyond the max. value of 132.
Prepr. action :It assumes 132.
User action : Correct it.

ILLEGAL OPTION: text E

Explanation : text is an unidentified option, e. g. OPTSET(NOPROPER);
Prepr. action : The option is ignored.
User action : Correct the error or remove the appropriate OPTSET statement at all.

TEXT BEGINNING 'text' HAS BEEN
DELETED DUE TO AN ERROR IN LINE xxxxx

Explanation : The preprocessor informes you that some text has been deleted. Up to eight significant characters are printed out.
User action : Correct the error which had been detected and diagnosed beforehand.

An abnormal end with user code 2424.

Explanation : An implementation restriction has been violated. At present the number of significant characters within a FORMAC statement and the total length of a literal constant is restricted to a fixed value.

Note: There is no restriction upon the length of a comment, and upon the number of input cards, literal constants, and comments.

User action : Segment the FORMAC statement or literal constant printed out last during compilation.

```
INPUT TO KFA FORMAC PREPROCESSOR - LEVEL 0

 1        /* "BERN" EVALUATES THE BERNOULLI NUMBERS FROM 1 TO M  */
 2      BERN:PROC OPTIONS(MAIN);
 3              FORMAC_OPTIONS;
 4              OPTSET(NOEXPND);
 5
 6              M=100;
 7              LET( S=N );
 8              DO K=0 TO M;
 9              FIXEDA( K=K );
10                    DO I=K+1 BY -1 TO 1;
11                    FIXEDA( I=I );
12              LET( S=REPLACE(S,N**I,X**(I+1)/(I+1)) );
13                    END;
14          LET(  S=REPLACE(S,X,N);
15                   PHI=DIST((K+1)*S) ;
16                   B=1-EVAL(PHI,N,1) ;
17                   S=PHI+N*B );
18              LET(K=K+1);
19              PRINT_OUT(K;B);  PUT FILE(SYSPRINT) SKIP(3);
20              END;
21      END BERN;
```

```
  /* "BERN" EVALUATES THE BERNOULLI NUMBERS FROM 1 TO M  */                1
 BERN:PROC OPTIONS(MAIN);                                                  2
/* DECLARES FOR RUN-TIME ROUTINES.        */                              3
DECLARE                                                                   3
 DENFMC1 ENTRY(CHAR(*)),                                                  3
 DENFMC2 ENTRY(CHAR(*)),                                                  3
 DENFMC4 ENTRY(CHAR(*)),                                                  3
 DENFMC5 ENTRY,                                                           3
 DENFMC7 ENTRY(CHAR(*)),                                                  3
 DENFMC8 ENTRY(CHAR(*)),                                                  3
 DENFMC9 ENTRY(CHAR(*)) RETURNS(BIN FLOAT(53)),                           3
 DENFMCA ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           3
 DENFMCB ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           3
 DENFMCC ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           3
 DENFMCD ENTRY(CHAR(*),BIN FLOAT(53)),                                    3
 DENFMCE ENTRY(CHAR(*),BIN FIXED(31)),                                    3
 DENFMCF ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1)),                          3
 DENFMCG ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1));                          3
                                                                          3
CALL DENFMC3(1B,-11010110000001100111111110010118);                       4
        M=100;                                                            6
CALL DENFMC1('S=N');                                                      7
        DO K=0 TO M:                                                      8
CALL DENFMCE('K=31622',(K));                                              9
             DO I=K+1 BY -1 TO 1;                                        10
CALL DENFMCE('I=31622',(I));                                             11
CALL DENFMC1('S=REPLACE(S,N**I,X**(I+1)/(I+1))');                        12
             END;                                                        13
  DO;                                                                    14
CALL DENFMC1('S=REPLACE(S,X,N)');                                        14
CALL DENFMC1('PHI=DIST((K+1)*S)');                                       15
CALL DENFMC1('B=1-EVAL(PHI,N,1)');                                       16
CALL DENFMC1('S=PHI+N*B');                                               17
  END;                                                                   17
CALL DENFMC1('K=K+1');                                                   18
  DO;                                                                    19
CALL DENFMC2('K');                                                       19
CALL DENFMC2('B');                                                       19
  END;                                                                   19
                        PUT FILE(SYSPRINT) SKIP(3);                      19
        END;                                                             20
 END BERN;                                                               21
```

```
I = 2
-----
S =  - 1/2 COS ( 2 X ) + 1/2
----------------------------
T = 1/4
-------

EX='S = -COS(X*2)*(1/2 ) + (1/2 )';


I = 3
-----
S = 3/4 SIN ( X ) - 1/4 SIN ( 3 X )
-----------------------------------
T = 1/8
-------

EX='S = SIN(X)*(3/4 )-SIN(X*3)*(1/4 )';


I = 4
-----
S =  - 1/2 COS ( 2 X ) + 1/8 COS ( 4 X ) + 3/8
----------------------------------------------
T = 1/16
--------

EX='S = -COS(X*2)*(1/2 ) + COS(X*4)*(1/8 ) + (3/8 )';


I = 5
-----
S = 5/8 SIN ( X ) - 5/16 SIN ( 3 X ) + 1/16 SIN ( 5 X )
-------------------------------------------------------
T = 1/32
--------

EX='S = SIN(X)*(5/8 )-SIN(X*3)*(5/16 ) + SIN(X*5)*(1/16 )';


I = 6
-----
S =  - 15/32 COS ( 2 X ) + 3/16 COS ( 4 X ) - 1/32 COS ( 6 X ) + 5/16
---------------------------------------------------------------------
T = 1/64
--------

EX='S = -COS(X*2)*(15/32 ) + COS(X*4)*(3/16 )-COS(X*6)*(1/32 ) + (5/16 )';


I = 7
-----
S = 35/64 SIN ( X ) - 21/64 SIN ( 3 X ) + 7/64 SIN ( 5 X ) - 1/64 SIN ( 7 X )
------------------------------------------------------------------------------
T = 1/128
---------

EX='S = SIN(X)*(35/64 )-SIN(X*3)*(21/64 ) + SIN(X*5)*(7/64 )-SIN(X*7)*(1/64 )';
```

```
INPUT TC KFA FORMAC PREPROCESSOR - LEVEL 0

 1      TRIGCM:PROC OPTIONS(MAIN);
 2             FORMAC_OPTIONS;
 3             OPTSET(EXPND;TRANS);
 4             OPTSET(ERSNP);
 5
 6             N=10;
 7             LET( S=SIN(X) );
 8             DO I=2 TO N;          LET( I="I" );
 9               LET( S=DIST( S * SIN(X) ) );
10                  DO K=1 TO I; LET(K="K");
11                    LET(S=REPLACE(S,
12                  SIN(X)*SIN(K*X),1/2*COS((1-K)*X)-1/2*COS((1+K)*X),
13                  SIN(X)*COS(K*X),1/2*SIN((1-K)*X)+1/2*SIN((1+K)*X)) );
14                  END;
15             PRINT_OUT(I;S);
16             LET( T=EVAL(S,X,#P/6) );
17             PRINT_OUT(T);
18
19       DCL EX CHAR(10000) VAR;  CHAREX(EX=S);  PUT DATA(EX) SKIP;
20             PUT SKIP(3);
21             END;
22
23       END TRIGCM;
```

```
PROC OPTIONS(MAIN);                                                    1

 TRIGOM:PROC OPTIONS(MAIN);                                            1
/* DECLARES FOR RUN-TIME ROUTINES.          */                        2
DECLARE                                                               2
 DENFMC1 ENTRY(CHAR(*)),                                              2
 DENFMC2 ENTRY(CHAR(*)),                                              2
 DENFMC4 ENTRY(CHAR(*)),                                              2
 DENFMC5 ENTRY,                                                       2
 DENFMC7 ENTRY(CHAR(*)),                                              2
 DENFMC8 ENTRY(CHAR(*)),                                              2
 DENFMC9 ENTRY(CHAR(*)) RETURNS(BIN FLOAT(53)),                       2
 DENFMCA ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                       2
 DENFMCB ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                       2
 DENFMCC ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                       2
 DENFMCD ENTRY(CHAR(*),BIN FLOAT(53)),                                2
 DENFMCE ENTRY(CHAR(*),BIN FIXED(31)),                                2
 DENFMCF ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1)),                      2
 DENFMCG ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1));                      2
                                                                      2
  DO;                                                                 3
CALL DENFMC3(1B,-110100111111100001111111001011B);                    3
CALL DENFMC3(1B,-110100111110111001111111001011B);                    3
  END;                                                                3
CALL DENFMC3(1B,-110100111101111001111111010110B);                    4
        N=10;                                                         6
CALL DENFMC1('S=SIN(X)');                                             7
        DO I=2 TO N;                                                  8
CALL DENFMC1('I=('||I||')');                                          8
CALL DENFMC1('S=DIST(S*SIN(X))');                                     9
            DO K=1 TO I;                                             10
CALL DENFMC1('K=('||K||')');                                         10
CALL DENFMC1('S=REPLACE(S,SIN(X)*SIN(K*X),1/2*COS((1-K)*X)-1/2*COS((1+K)*X   13
),SIN(X)*COS(K*X),1/2*SIN((1-K)*X)+1/2*SIN((1+K)*X))');              13
            END;                                                     14
  DO;                                                                15
CALL DENFMC2('I');                                                   15
CALL DENFMC2('S');                                                   15
  END;                                                               15
CALL DENFMC1('T=EVAL(S,X,#P/6)');                                    16
CALL DENFMC2('T');                                                   17
  DCL EX CHAR(10000) VAR;                                            19
CALL DENFMCH(EX,'S');                                                19
                                        PUT DATA(EX) SKIP;           19
        PUT SKIP(3);                                                 20
        END;                                                         21
  END TRIGOM;                                                        23
```

```
        INPUT TO KFA FORMAC PREPROCESSOR - LEVEL 0

     1                  /*  SAMPLE  */
     2        FEATURE:PROC OPTIONS(MAIN);
     3
     4             /*  THIS SAMPLE SHOWS YOU SOME OF
     5                 THE ADVANTAGES OF THE
     6                   KFA FORMAC PREPROCESSOR:
     7
     8                       INTERSPERSED MESSAGES,
     9                       NO COMPRESSION OF STATEMENTS, LITERAL CONSTANTS,
    10                         ETC.
    11                       BACKWARD REFERENCE,
    12                       " PERMITTED IN ALL MACROS AND FUNCTIONS,
    13                       A MORE FLEXIBLE INTERPRETATION OF 'CONVERT',
    14                       BETTER DIAGNOSTICS.
    15
    16             <NOTE THAT THIS COMMENT IS LEFT UNCHANGED>
    17             */
    18        DECLARE PSTRING CHAR(80), BIT BIT(1),  Z(3);
    19
    20        LOOP:GET FILE(SYSIN) EDIT(PSTRING) (A(80));
    21
    22             FORMAC_OPTIONS;
    23             OPTSET( EDIT, LINELENGTH=72;IMPROPER; ERRSNAP);
INVALID DELIMITER. ';' ASSUMED
ILLEGAL OPTION: ERRSNAP
    24
    25             CONVERT FLOAT(X;Y;Z(1))
ILLEGAL VAR. IN CONVERT STMT
TEXT BEGINNING '(1' HAS BEEN DELETED DUE TO AN ERROR IN LINE    25
    26
    27                       FIXED(I;K;J) FIXED(L;M;K;  T12345678) ;
DOUBLE DECL. OF K IN CONVERT STMT
T12345678 TOO LONG
    28
    29         LET(  "I" = "A" + "B" +C("K","J"+K) + C("L",1)  );
    30
    31         LET( "X" = B(9,1 ;
RIGHT PAREN(S) INSERTED
    32         SAVE( A=B(10,1)));
RIGHT PAREN IGNORED
    33
    34        B1:BEGIN;    FORMAC_OPTIONS(PRINT);
UNEXPECTED KEYWORD
TEXT BEGINNING 'FORMAC_O' HAS BEEN DELETED DUE TO AN ERROR IN LINE    34
    35         LET(  DS = (F.($(1)))**ZX + B(1);
    36               /* COMMENTS ARE ALLOWED WITHIN MACROS, ETC. */
    37               DB=DS 2 );
INVALID: DB=DS 2
:PAREN MAY BE MISSING
    38
    39        END B1;
    40         L=LOP(A) + LOP(C) - INTEGER(7);
    41         IF IDENT(A;B) THEN K=LOP(A);
    42                       ELSE  K=LOP(B);
    43         FORM(PSTRING; PSTRING ),
INVALID DELIMITER. ';' ASSUMED
    44         /* SET ERROR SNAP ON */        OPTSET(ERSNP);
    45           CALL DATAIN( NARGS(VAR),(PSTRING));
    46
    47       END FEATURE;
```

```
            /*  SAMPLE  */                                                  1
  FEATURE:PROC OPTIONS(MAIN);                                              2
       /*  THIS SAMPLE SHOWS YOU SOME OF                                   4
           THE ADVANTAGES OF THE                                           5
             KEA FORMAC PREPROCESSOR:                                      6
                                                                           7
                 INTERSPERSED MESSAGES,                                    8
                 NO COMPRESSION OF STATEMENTS, LITERAL CONSTANTS,          9
                   ETC.                                                   10
                 BACKWARD REFERENCE,                                      11
                 " PERMITTED IN ALL MACROS AND FUNCTIONS,                 12
                 A MORE FLEXIBLE INTERPRETATION OF 'CONVERT',             13
                 BETTER DIAGNOSTICS.                                      14
                                                                          15
       <NOTE THAT THIS COMMENT IS LEFT UNCHANGED>                         16
       */                                                                 17
  DECLARE PSTRING CHAR(80), BIT BIT(1),  Z(3);                            18
  LOOP:GET FILE(SYSIN) EDIT(PSTRING) (A(80));                             20
/* DECLARES FOR RUN-TIME ROUTINES.          */                           22
DECLARE                                                                   22
 DENEMC1 ENTRY(CHAR(*)),                                                  22
 DENEMC2 ENTRY(CHAR(*)),                                                  22
 DENEMC4 ENTRY(CHAR(*)),                                                  22
 DENEMC5 ENTRY,                                                           22
 DENEMC7 ENTRY(CHAR(*)),                                                  22
 DENEMC8 ENTRY(CHAR(*)),                                                  22
 DENEMC9 ENTRY(CHAR(*)) RETURNS(BIN FLOAT(53)),                           22
 DENEMCA ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           22
 DENEMCB ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           22
 DENEMCC ENTRY(CHAR(*)) RETURNS(BIN FIXED(31)),                           22
 DENEMCD ENTRY(CHAR(*),BIN FLOAT(53)),                                    22
 DENEMCE ENTRY(CHAR(*),BIN FIXED(31)),                                    22
 DENEMCF ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1)),                          22
 DENEMCG ENTRY(CHAR(*),CHAR(*)) RETURNS(BIT(1));                          22
                                                                          22
  DO;                                                                     23
CALL DENEMC3(1B,-110100101111111001111111110101108);                      23
CALL DENEMC3(1B,+100000000000000010010000010111108);                      23
CALL DENEMC3(1B,-110101101000000001111111110101108);                      23
  END;                                                                    23
  DO;                                                                     29
CALL DENEMCE('K=31622',(K));                                              29
CALL DENEMCE('J=31622',(J));                                              29
CALL DENEMCE('L=31622',(L));                                              29
CALL DENEMC1('I=('||A||')+('||B||')+C(K,J+K)+C(L,1)');                    29
I=DENEMCA('I');                                                           29
  END;                                                                    29
  DO;                                                                     31
CALL DENEMC1('X=B(9,1)');                                                 31
X=DENEMC9('X');                                                           31
  END;                                                                    31
CALL DENEMC4('A=B(10,1)');                                                32
  B1:BEGIN;                                                               34
  DO;                                                                     35
CALL DENEMC1('DS=(E.(*(1)))**7X+B(1)');                                   35
        /* COMMENTS ARE ALLOWED WITHIN MACROS, ETC. */                    36
CALL DENEMC1('DB=DS 2');                                                  37
  END;                                                                    37
                  );                                                      37
  END B1;                                                                 39
   L=DENEMCB('A') + DENEMCB('C') - DENEMCA('7');                          40
   IF DENEMCG('79999997=A','79999998=B') THEN K=DENEMCB('A');             41
```

```
                ELSE  K=DENFMCB('B');                                    42
  DO;                                                                    43
CALL DENFMC1(PSTRING);                                                   43
CALL DENFMC1(PSTRING);                                                   43
  END;                                                                   43
   /* SET ERROR SNAP ON */                                               44
CALL DENFMC3(18,-11010011110111100111111110101108);                      44
     CALL DATAIN( DENFMCC('VAR'),(PSTRING));                             45
 END FEATURE;                                                            47
```