✓
ANL-PHY-76-3
September 1976

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

MASTER

USING MORTRAN TO TRANSLATE FORTRAN PROGRAMS
FROM ONE MACHINE TO ANOTHER^{*}

by

Steven C. Pieper
Physics Division

The author has a limited number of
copies for general distribution.
Anyone who desires a copy should
contact him directly.

* Work performed under the auspices of the U.S. Energy Research and
Development Administration.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Energy Research and Development Administration, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona	Kansas State University	The Ohio State University
Carnegie-Mellon University	The University of Kansas	Ohio University
Case Western Reserve University	Loyola University	The Pennsylvania State University
The University of Chicago	Marquette University	Purdue University
University of Cincinnati	Michigan State University	Saint Louis University
Illinois Institute of Technology	The University of Michigan	Southern Illinois University
University of Illinois	University of Minnesota	The University of Texas at Austin
Indiana University	University of Missouri	Washington University
Iowa State University	Northwestern University	Wayne State University
The University of Iowa	University of Notre Dame	The University of Wisconsin

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights. Mention of commercial products, their manufacturers, or their suppliers in this publication does not imply or connote approval or disapproval of the product by Argonne National Laboratory or the U. S. Energy Research and Development Administration.

Using Mortran to Translate Fortran Programs
From One Machine to Another

Steven C. Pieper

September 29, 1976

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Physics Division
Argonne National Laboratory
Argonne, Illinois 60439

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Abstract

The Mortran processor was slightly modified, and a new set of macros written, to act as an aid in the translation of IBM Fortran programs to CDC Fortran. A 30,000 source card program was successfully converted using the resulting package. The capabilities and limitations of this approach to Fortran conversion are described and two examples of the output of the package are given.

This summer Malcolm Macfarlane and I had occasion to translate a large (30,000 source cards) Fortran program from IBM Fortran to CDC (specifically BKY FTN 4) Fortran. Of principal interest was that the translated version not become a dead end, but that as future improvements to the IBM version were made, they could easily and automatically be incorporated into the CDC version. It was decided to use Mortran* as an aid in the translation. Since our aims were somewhat different from those of the authors of Mortran, several changes to the Mortran processor had to be made. These, all rather superficial, were:

- 1) Control cards (%S and %O) were added to change from Mortran-style input to standard Fortran card-image input and back again. In Fortran image input mode, columns 1-5 contain statement numbers, and column 6 is a continuation indicator. A "C" in column 1 designates a comment card. Comments are copied to the output and Fortran statements are made acceptable to Mortran by removing continuation characters and statement numbers and adding semicolons. The statement numbers are saved and inserted back into the Mortran output as it is generated. Note that these new options are different from the Mortran mode switches of %F and %M. This simple addition meant that no changes had to be made to the existing 30,000 source cards to make them acceptable to Mortran.

- 2) The processor was changed to read two cards that contained two sets of three-character flags. The first set of flags mark cards that are to be included in the source -- any card having one of these flags in columns 1 to 3 will be processed by having the flag removed and the rest of the card shifted left three columns. The second set of flags marks cards that are to be deleted from the source as it is read.

- 3) The standard Mortran macros were removed (to conserve space in the macro table) and a new set of macros was made for the translation. This set consists of general conversion aids and some conversions that are specific to the program that was being translated.

The modified Mortran is capable of automatically making the following IBM to CDC conversions:

- 1) All standard double precision function names are converted to the corresponding single precision name.

- 2) Double precision constants that have one of the forms

* A User's Guide to MORTRAN2, A. J. Cook and L. J. Shustek, Computation Research Group, Stanford Linear Accelerator Center, CGTM No. 165, 1975.

.d0Deee, d.d0Deee, or d.Deee where d represents a string of digits and eee is an exponent are converted to single precision constants.

3) IMPLICIT type*length (...) statements are converted by removing the "*length".

4) type*length statements are converted by removing the "*length", except that LOGICAL*1 statements are converted to INTEGER statements since they are usually used for character manipulation in the program that was being translated.

5) Type statements that contain data initialization values are broken into separate type and DATA statements.

6) All DATA statements are collected and placed following all other specification statements. [This requires the aid of a pseudo-Fortran statement (NOW) at the appropriate spot in the source deck.]

7) "END = statement number" phrases in READ statements are converted into separate
IF (EOF(unit) .NE. 0) GO TO statement number
statements.

8) Argument lists are removed from ENTRY statements.

9) RETURN i statements are converted to RETURN ITRNi statements. However, the necessary modifications of SUBROUTINE statements that contain RETURN points must be made manually.

10) Statements of the form

CALL ... (... , &statement number ...)

are converted to the corresponding CDC form:

CALL ... (....), RETURNS (statement number)

11) The following Fortran-H Extended-Logic functions are converted to the corresponding CDC inline functions:

A) TBIT functions that check if a number is even or odd.

B) LAND and LOR functions.

C) SHFTR and SHFTL functions.

12) A set of macros for collecting characters into eight-character names is provided. However, these require explicit invocation in the source.

Once this conversion package was ready, the translation effort consisted of preparing the source deck for the conversion and of writing specific macros for the program. Examples of the steps needed to prepare the deck are the placing of NOW statements at the end of the other specification statements, putting double precision

constants in the required form, providing separate IBM and CDC versions of SUBROUTINE statements that contain RETURN points, and providing separate IBM and CDC versions of DATA statements that contain hexadecimal values. The different versions of such cards are prepared with the keys CIB or CDC in columns 1 to 3, and are then selected according to which version is being generated. Specific macros were written to convert the argument lists of references to ENTRY points that (in the IBM version) have a different argument list from the primary entry point. When this preparation was complete, we had a single source deck that could be used to automatically prepare either the CDC or the IBM version of the program. Approximately three man-weeks were spent preparing the source deck, getting the CDC version of the program to compile, and verifying that the CDC version executed correctly.

The principal advantage of this approach to converting Fortran source programs is that one is left with a single Fortran source deck. Future development is made using this deck and from time to time one can automatically generate updated versions of the program for other computers. There are, however, also other advantages. Mortran is written in ANSI-standard Fortran and thus will compile and execute on almost any large computer. The above mentioned translation was begun on the Argonne IBM computer system and was finished on the Berkeley CDC computers. Development of the program, and future translations, will now be done on the Argonne computers. Mortran processes statements (with few exceptions) one at a time, and thus does not require large amounts of core or CPU time. When the Mortran processor is compiled with an optimizing compiler, one achieves conversion times that are comparable to the subsequent compilation times with the same compiler (with the IBM compilers the times were nearly equal; for the CDC compiler the translation time was about four times longer than the compilation time).

Another advantage is that one can use Mortran for purposes other than the translation. A set of macros defining the common blocks used by the program has been written and specific common blocks are included in a source module by a card that simply names the common block. This means that there is only one version of each common block that must be maintained and updated as changes are made to the program.

Disadvantages of this conversion package are generally associated with the difficulty one has in limiting the scope of a Mortran substitutional parameter. As an example, double precision numbers are recognized by the presence of strings such as `OD0`, `OD+`, or `OD-`. This means both that many double precision numbers will not be converted, and that some undesired conversion will occur (for example, `AOD+B` is converted to `AOE+B`). Some difficulty was encountered in processing literal DATA values, and strings of several blanks in such literals are reduced to a single blank. However, literals in FORMAT statements are properly processed (providing they are delimited by apostrophies).

The remainder of this report consists of some examples. First the macros for producing IBM and CDC versions are given (for brevity most of the common block definitions have been left out). Then two subroutines in Mortran input form and in both IBM and CDC translated form are shown. The first subroutine is really just a collection of typical Fortran statements that must be converted to be acceptable to the CDC compilers. The second subroutine is the basic input processor for the program that was translated. It carries out extensive character manipulation and thus is more machine dependant than the computational sections of the program. Therefore an unusually large number of specific references to the Mortran macros had to be added to the source.

IBM Macros

The following macros are used when an IBM version of the source is desired. The first card contains the keys that specify source cards that are to be included in the output version (by having the key stripped off the card). The second card contains the keys that specify source cards that are to be excluded from the output. The third card specifies the Mortran character set.

```
CIB CIE                                IBM VERSION
CDE
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ .(),=+*/ <>:;' " %#@[&?
'.@.#.NXPTRPCS% '.#.'.= '.#.'.@.M G NXPTRPCS% '.#.'.= * '.#.'.@.M C
"
```

FOLLOWING ARE THE MACROS THAT DEFINE THE COMMON BLOCKS.
THEY ARE THE SAME IN BOTH THE CDC AND IBM VERSIONS

```
"
%';ALLOCS;' = ' ;
    INTEGER*4 Z ;
    COMMON /LOC/ Z(75) ;
    INTEGER FACFR4, FACFR2, FACFR1 ;
    COMMON /ALLOCS/ FACFR4, FACFR2, FACFR1 ;
    DIMENSION ALLOC(1) ;
    LOGICAL*1 MLLOC(1) ;
    REAL*4 ALLOC4(1) ;
    INTEGER ILLOC(1) ;
    EQUIVALENCE ( FACFR4, ALLOC(1), ILLOC(1), ALLOC4(1), MLLOC(1) ) ;
    ALLOCLEVEL ALLOC, MLLOC, ALLOC4, ILLOC, FACFR4, FACFR2,
    FACFR1 ;
;
%';SWITCH;' = ' ;
    COMMON /SWITCH/ IASYMP, ICHECK, IECHO, IELAST, IFIT,
    IVRTEX, ISAVE, ITSO, LINEAR, NUCONL,
    NEXT, IFITER, IEPOW,
    ISW1, ISW2, ISW3
;
%';INPBUF;' = ' ; LOGICAL*1 IBUF(80) ;
    COMMON /INPBUF/ NOCH, INCH, IBUF;
"
```

FOLLOWING ARE THE IBM MACROS

```
%';IMPLICIT;' = ' ; IMPLICIT REAL*8 ( A-H, O-Z ) ; GENERIC ;'
```

```
%';ALLOCLEVEL#;' = ' ;'
```

" MACROS TO GET INDIVIDUAL CHARACTERS "

```
%'CHARACTERVALUE(#)' = '#1'
%'CHARDATAVALUE<#>' = '1H#1'
%'CHARDATAVALUE''#'' = '1H#1'
%'CHARTONUM(#)' = 'CHARACTERVALUE(#1) - 240'
```

" MACROS TO COLLECT CHARACTERS INTO 8-CHARACTER NAMES "

```
%';WORDINIT;' = ';REAL*8 WORD, BLANK8 /' ' ' /;  
  LOGICAL*1 WORD1(8);  
  EQUIVALENCE ( WORD, WORD1(1) );'  
%';WORDSTART;' = ';WORD = BLANK8; IWORD = 0;'  
%';WORDADD(#);' = ';IF (IWORD .LT. 8) WORD1(IWORD+1) = #1;  
  IF (IWORD .LT. 8) IWORD = IWORD+1;'  
%';WORDSAVE(#);' = ';#1 = WORD;'  
  
%';NOW;' = ';
```

" FOLLOWING ARE SO THAT FIRST STATEMENT IS PRECEDED
 BY A SEMICOLON "

```
:::  
%S  
%U5  
%%
```

CDC Macros

The following macros are used when a CDC version of the source is desired.

CDC CDE CDS SMALL-CORE MEMORY VERSION

CIE

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ . () , = + - * / < > ; : ' " % # @ [& 0
' . @ . # . NXPTRPCS% ' . # . ' . = ' . # . ' . @ . M G NXPTRPCS% ' . # . ' . = * ' . # . ' . @ . M C
"

FOLLOWING ARE THE MACROS THAT DEFINE THE COMMON BLOCKS
THEY ARE THE SAME IN BOTH THE CDC AND IBM VERSIONS

```
"
%';ALLOCS;' = ' ;
    INTEGER*4 Z ;
    COMMON /LOC/ Z(75) ;
    INTEGER FACFR4, FACFR2, FACFR1 ;
    COMMON /ALLOCS/ FACFR4, FACFR2, FACFR1 ;
    DIMENSION ALLOC(1) ;
    LOGICAL*1 MLLOC(1) ;
    REAL*4 ALLOC4(1) ;
    INTEGER ILOC(1) ;
    EQUIVALENCE ( FACFR4, ALLOC(1), ILOC(1), ALLOC4(1), MLLOC(1) ) ;
    ALLOCLEVEL ALLOC, MLLOC, ALLOC4, ILOC, FACFR4, FACFR2,
    FACFR1 ;
```

```
;' ;SWITCH;' = ' ;
    COMMON /SWITCH/ IASYMP, ICHECK, IECHO, IELAST, IFIT,
    IVRTEX, ISAVE, ITSO, LINEAR, NUCONL,
    NEXT, IFITER, IEPOW,
    ISW1, ISW2, ISW3
```

```
;' ;INPBUF;' = ' ;LOGICAL*1 IBUF(80) ;
    COMMON /INPBUF/ NOCH, INCH, IBUF;'
```

FOLLOWING ARE THE CDC MACROS

```
%';ALLOCLEVEL' = ' ;LEVEL 1, '
```

```
%';IMPLICIT;' = ' ; ;'
%';IMPLICIT###(' = ' ;IMPLICIT #1 ('
```

CONVERT DOUBLE PRECISION FUNCTIONS TO SINGLE PRECISION

```
%';TOSINGLE #;' = ' %' 'D#1(' ' = ' ' #1(' ' ;'
;TOSINGLE EXP ;
TOSINGLE SQRT ;
TOSINGLE ATAN ;
TOSINGLE ATAN2 ;
TOSINGLE SIN ;
```



```

TOSINGLE COS      ;
TOSINGLE TAN      ;
TOSINGLE SINH     ;
TOSINGLE COSH     ;
TOSINGLE ABS      ;
TOSINGLE GAMMA    ;
TOSINGLE FLOAT    ;
TOSINGLE SIGN     ;
TOSINGLE REAL     ;
TOSINGLE CMPLX    ;
TOSINGLE CONJG    ;

```

```

%';TOASINGLE#;' = ' %'D#1(''='A#1('' ;'

```

```

TOASINGLE LOG      ;
TOASINGLE LGAMA    ;
TOASINGLE MAX1     ;
TOASINGLE MIN1     ;
TOASINGLE MOD      ;
TOASINGLE INT      ;
TOASINGLE IMAG     ;
%'DARCOS(' = 'ACOS('
%'CDABS(' = 'CABS('

```

```

%';CALLSTIMER;' = ';;'
%'TTIMER(' = 'SECOND('
%'LOOKDD(#' = '1'

```

" FORTRAN H XL FUNCTIONS - TBIT USED TO CHECK FOR EVEN/ODD "

```

%'TBIT( #,30)' = '( (IABS(#1) .AND. 2B) .EQ. 2B )'
%'TBIT( #,31)' = '( (IABS(#1) .AND. 1B) .EQ. 1B )'
%'LAND( #, #)' = '( #1 .AND. #2 )'
%'LOR( #, #)' = '( #1 .OR. #2 )'
%'SHFTR( #, #)' = 'SHIFT( #1, -#2 )'
%'SHFTL( #, #)' = '( SHIFT( #1, #2 ) .AND. MASK(60-#2) )'

```

" MACROS TO DEFINE CHARACTERS "

```

%'CHARACTERVALUE( #)' = '(SHIFT( #1,6) .AND. 77B)'
%'CHARDATAVALUE<#>' = '1R#1'
%'CHARDATAVALUE' '#' = '1R#1'
%'CHARTONUM( #)' = 'CHARACTERVALUE( #1) - 1R0'

```

" CONVERT STATEMENT NUMBERS IN CALLS AND RETURNS

```

%';RETURN#;' = ';"RETURN" ITRN#1;'
%';RETURN;' = '";RETURN;"'
%')RETURN#;' = ')"RETURN" ITRN#1;'
%')RETURN;' = ')"RETURN";'
%';CALL#( #,&#);' = ';"CALL #1 ( #2 ), RETURNS ( #3 );'
%';CALL#( #,&#,#);' = ';"CALL #1 ( #2, #4 ), RETURNS ( #3 );'
%';CALL#( #,&#),RETURNS( #);' = ';"CALL #1 ( #2 ), RETURNS ( #4, #3 );'
%';CALL#( #,&#,#),RETURNS( #);' = ';"CALL #1 ( #2, #4 ), RETURNS ( #5, #3 );'

```

```
%';ENTRY#( # );' = ';ENTRY #1;'  
%';SUBROUTINE#( #/# );' = ';SUBROUTINE #1 ( #2 #3 );'  
%';FUNCTION#( #/# );' = ';FUNCTION #1 ( #2 #3 );'
```

" CONVERT SOME DOUBLE PRECISION NUMBERS TO SINGLE PRECISION

```
%'OD0' = 'OE0'  
%'OD+' = 'OE+ '  
%'OD-' = 'OE- '  
%'.D' = '.E'  
%';FORMAT( #D' = ';FORMAT ( #1E'  
%';FORMAT( # ( #D' = ';FORMAT ( #1( #2D'
```

" IN THE ABOVE DO NOT BE CONFUSED BY LOGICAL DOTS

```
%';LEAVELOGICAL#;' = '%'.#1.' = '.".#1."' ; '
```

```
;LEAVELOGICAL EQ ;  
LEAVELOGICAL LE ;  
LEAVELOGICAL GE ;  
LEAVELOGICAL NE ;  
LEAVELOGICAL LT ;  
LEAVELOGICAL GT ;  
LEAVELOGICAL AND ;  
LEAVELOGICAL OR ;  
LEAVELOGICAL NOT ;
```

" CONVERT END-OF-FILE BRANCHES AND DELETE ERROR BRANCHES "

```
%'READ( #,END=# ) #;' = 'READ( #1 ) #3;IF ( EOF( #1 ) .NE. 0 ) GO TO #2;'  
%'READ( #, #,END=# ) #;' = 'READ( #1, #2 ) #4;IF ( EOF( #1 ) .NE. 0 ) GO TO #3;'  
%'READ( #,ERR=# )' = 'READ( #1 )'  
%'READ( #,ERR= #, ' = 'READ( #1, '  
%'WRITE( #,ERR=# ) #' = 'WRITE( #1 ) #3'
```

" CONVERT TYPE STATEMENTS AND SPLIT OFF DATA INITIALIZATION "

```
%';INTEGER#/#/;' = ';INTEGER #1;DATA #1 /#2/;'  
%';INTEGER#/#/, #;' = ';INTEGER #1;DATA #1 /#2/;INTEGER #3;'  
%';INTEGER#( # )/#/;' = ';INTEGER #1( #2 );DATA #1 /#3/;'  
%';INTEGER#( # )/#/, #;' = ';INTEGER #1( #2 );DATA #1 /#3/;INTEGER #4;'  
%';INTEGER#, #/#/' = ';INTEGER #1;INTEGER #2 /#3/ '  
%';INTEGER#/#/, ' = ';INTEGER #1 /#2/; INTEGER '  
%';INTEGER*2' = ';INTEGER '  
%';INTEGER*4' = ';INTEGER '
```

```
%';REAL#/#/;' = ';REAL #1;DATA #1 /#2/;'  
%';REAL#/#/, #;' = ';REAL #1;DATA #1 /#2/;REAL #3;'  
%';REAL#( # )/#/;' = ';REAL #1( #2 );DATA #1 /#3/;'  
%';REAL#( # )/#/, #;' = ';REAL #1( #2 );DATA #1 /#3/;REAL #4;'  
%';REAL#, #/#/' = ';REAL #1;REAL #2 /#3/ '  
%';REAL#/#/, ' = ';REAL #1 /#2/; REAL '  
%';REAL*8' = ';REAL '  
%';REAL*4' = ';REAL '
```

```
%';LOGICAL#/#/;' = ';LOGICAL #1; DATA #1 /#2/;'
%';LOGICAL#/#/,#;' = ';LOGICAL #1; DATA #1 /#2/; LOGICAL #3;'
%';LOGICAL#,#/#/' = ';LOGICAL #1;LOGICAL #2 /#3/'
%';LOGICAL#/#/, ' = ';LOGICAL #1 /#2/; LOGICAL'
```

```
%';COMPLEX*16' = ';COMPLEX'
%';COMPLEX*8' = ';COMPLEX'
%';LOGICAL*1' = ';INTEGER'
%';LOGICAL*4' = ';LOGICAL'
```

" SAVE ALL DATA STATEMENTS UNTIL A NOW CARD IS READ "

```
%';CLEARDATA;' = '%';DA1;'='';' %';DA2;'='';'
%';DA3;'='';' %';DA4;'='';' %';DA5;'='';'
%';DA6;'='';' %';DA7;'='';' %';DA8;'='';'
%';DA9;'='';' %';DA10;'='';' %';DA11;'='';'
%';DA12;'='';' %';DA13;'='';' %';DA14;'='';'
%';DA15;'='';' %';DA16;'='';' %';DA17;'='';'
%';DA18;'='';' %';DA19;'='';' %';DA20;'='';'
%';DA21;'='';'
%';TEMPDATA##;'='';(D1)##1;' ;
%';NOW;' = ';DA1;DA2;DA3;DA4;DA5;DA6;DA7;DA8;DA9;DA10;DA11;
DA12;DA13;DA14;DA15;DA16;DA17;DA18;DA19;DA20;DA21;
CLEARDATA;'
;CLEARDATA;
%';DEFD#,#;' = '%';(D#1)##;'='';%';DA#1;'='';OUTDATA##1;'
%';TEMPDATA####;'='';(D#2)####1;' ;
;DEFD1,2;DEFD2,3;DEFD 3,4; DEFD 4,5; DEFD 5,6; DEFD 6,7;
DEFD 7,8; DEFD 8,9; DEFD 9,10; DEFD 10,11; DEFD 11,12;
DEFD 12,13; DEFD 13,14; DEFD 14,15; DEFD 15,16; DEFD 16,17;
DEFD 17,18; DEFD 18,19; DEFD 19,20; DEFD 20,21; DEFD 21,22;
%';INDATA#;' = ';TEMPDATA#1;'
%';INDATA#'#'#;' = ';INDATA#1<#2>#3;'
%';DATA' = ';INDATA'
%';OUTDATA#;' = ';DATA "#1;'
%';OUTDATA#<#>#;' = ';OUTDATA#1'#2'#3;'
%';OUTDATA#<>#;' = ';OUTDATA#1' '#2;'
```

" COLLECT CHARACTERS INTO EIGHT-CHARACTER WORDS "

```
%';WORDINIT;' = ';REAL*8 WORD, BLANK8 /' ' /;'
%';WORDSTART;' = ';WORD = BLANK8; IWORD = 0;'
%';WORDADD(#);' = ';IF (IWORD .LT. 8)
WORD = (SHIFT(WORD,6) .AND. 777777777777777700B) .OR.
CHARACTERVALUE(#1) ;
IF (IWORD .LT. 8) IWORD = IWORD+1;'
%';WORDSAVE(#)' = ';IWORD = 10-IWORD;
IF (IWORD .GT. 0) WORD = SHIFT(WORD, 6*IWORD);
#1 = WORD;'
```

" MAKE NXINT, NXVAL, ETC, ARGUMENT LISTS COMPATABLE "

```
%';CALLNXINT(,#,&#,&#,&#);' = ';
```

```

CALL NXINT ( #1, DUMMY ), RETURNS ( #2, #3, #4, #2 );'
%';CALLNXVAL( #, &#, &#, &# );' = ' ;
CALL NXVAL ( IDUMMY, #1 ), RETURNS ( #2, #3, #4, #2 );'
%';CALLNXVALF( #, &#, &#, &#, &# );' = ' ;
CALL NXVALF ( IDUMMY, #1 ), RETURNS ( #2, #3, #4, #5 );'
%';CALLNXHINT( #, &#, &#, &# );' = ' ;
CALL NXHINT ( #1, DUMMY ), RETURNS ( #2, #3, #4, #2 );'
%';CALLNXWORD( #, &#, &#, &# );' = ' ;
CALL NXWORD ( IDUMMY, #1 ), RETURNS ( #2, #3, #4, #2 );'
"
MAKE OTHER ENTRY POINTS COMPATABLE WITH MAIN ENTRY POINT
"
%';CALLFITKEY( #, &#, &#, &#, &# );' = ' ;
CALL FITKEY ( IDUMMY, #1 ), RETURNS ( #2, #3, #4, #5 );'
%';CALLVCSQ12( # );' = ' ";CALL" VCSQ12 ( #1, IDUMMY, IDUMMY );'
%';NAMLOC( # )' = ' "NAMLOC( IDUMMY, #1 )" '
%';CALLPLSUB( #, #, # );' = ' ";CALL" PLSUB ( #1, IDUMMY, #2, #3 );'
%';CALLLIN( #, #, #, #, #, # );' = '
';"CALL" LIN ( #1, #2, #3, #4, #5, #1, #4, #6, #7 );'
" ENTER FORTRAN INPUT MODE AND GO TO UNIT 5 FOR THE DECK
TO BE CONVERTED "

```

```

%S
%U5
%%

```

SAMPLE - Mortran Input

The following is the Mortran input of the first example. This is not a functional subroutine, but is merely a collection of legal IBM Fortran statements that must be translated for CDC Fortran. Note the use of the three-letter keys in the first two cards to select the appropriate SUBROUTINE statement.

```
CIB      SUBROUTINE SAMPLE ( A, B, *, * )
CDC      SUBROUTINE SAMPLE ( A, B ), RETURNS ( ITRN1, ITRN2 )
C
      IMPLICIT REAL*8 ( A-H, O-Z )
C
      INTEGER*2  I / 12 /
      LOGICAL*1  CHAR /'A'/
C
      NOW
C
      X = DSIN( 5.30D0*B )
      IF ( TBIT(J,31) ) RETURN 2
      READ ( 5, 10, END=99, ERR=99 ) D
10      FORMAT ( D20.10 )
99      CALL Z ( D, &100, E, &200 )
100     J = LAND(J, 4)
200     J = SHFTL( J, 3 )
      RETURN 1
      END
```


SAMPLE - IBM and CDC Fortran

The following is the IBM Fortran version of the first example.

```

CDC      SUBROUTINE SAMPLE ( A, B, *, * )
C        SUBROUTINE SAMPLE ( A, B ), RETURNS ( ITRN1, ITRN2 )
C
C        IMPLICIT REAL*8 ( A-H, O-Z )
C
C        INTEGER*2 I / 12 /
C        LOGICAL*1 CHAR /      1HA/
C
C        X = DSIN( 5.30D0*B )
C        IF ( TBIT(J,31) ) RETURN 2
C        READ ( 5, 10, END=99, ERR=99 ) D
10       FORMAT ( D20.10 )
99       CALL Z ( D, &100, E, &200 )
100      J = LAND(J, 4)
200     J = SHFTL( J, 3 )
        RETURN 1
        END

```

The following is the CDC Fortran version of the first example.

```

CIB      SUBROUTINE SAMPLE ( A, B, *, * )
C        SUBROUTINE SAMPLE ( A, B ), RETURNS ( ITRN1, ITRN2 )
C
C        IMPLICIT REAL ( A-H, O-Z )
C
C        INTEGER I
C        INTEGER CHAR
C
C        DATA I /12/
C        DATA CHAR /      1HA/
C
C        X = SIN( 5.30E0*B )
C        IF ( ( IABS(J) .AND. 1B ) .EQ. 1B ) RETURN ITRN2
C        READ(5,10) D
C        IF ( EOF(5) .NE. 0 ) GO TO 99
10       FORMAT ( E20.10 )
99       CALL Z ( D, E ), RETURNS (100,200)
100      J = ( J .AND. 4 )
200     J = ( SHIFT(J,3) .AND. MASK(60-3) )
        RETURN ITRN1
        END

```

MSCAN - Mortran Input

The following is the Mortran input of the second example.
Note in particular the use of the Mortran macros for the collection of characters into words (statement number 200 and following).

```
SUBROUTINE MSCAN ( KEYEX, ITOKEN, VALU, IVALU, MESS,
1 IMESS, STOP )
```

```
FIELD SCANNER FOR DELPHI
MODIFIED MFIELD SCANNER OF SPEAKEZ
THIS VERSION IS FOR PTOLEMY.
```

```
THE LINE TO SCAN IS IN IBUF. IT CONTAINS NOCH CHARACTERS.
INCH IS THE FIRST CHARACTER TO CONSIDER AND IS UPDATED TO POINT
TO THE CHARACTER THAT STOPPED THE SCAN - I.E. TO THE FIRST
CHARACTER TO CONSIDER THE NEXT TIME.
```

```
ITOKEN IS SET AS FOLLOWS
```

```
> 0 CODE VALUE AS DESCRIBED SEVERAL LINES BELOW
```

- 0 NO FIELD
- 1 ALPHANUMERIC LEFT SHIFTED BLANK FILLED - STORRED IN 'VALU'.
THE NUMBER OF CHARACTERS IS STORED IN IMESS.
- 2 NUMERIC (FLT. PT. FORM) - STORRED IN 'VALU' AND 'IVALU'.
- 3 LEVEL SPEC (E.G. 1P3/2) LEFT SHIFTED BLANK FILLED -
STORRED IN 'VALU'. **** REMOVED - S.P. ****
- 4 LITERAL DATA (80 COL. MAX). STORRED IN MESS; IMESS IS NUM.
CHARACTERS STORED.
- 5 INTEGER (NO DECIMAL PT. OR E FIELD) - STORRED IN 'VALU' AND
'IVALU'.
- 8 END OF CARD BEFORE ANYTHING WAS FOUND - NOTHING STORED.
- 9 ERROR ENCOUNTERED - NOTHING RELIABLE STORED.

```
SPECIAL CHARACTER TOKENS ARE ( + - * ARE NO LONGER TOKENS )
```

```
% (1) # (2) & (3) : (4) ; (5)
- (6) + (7) / (8) * (9) ((10) (+ - NOW START NUMBERS)
(* NOW AN ALPHA CHAR.)
) (11) = (12) @ (13) < (14) > (15)
| (16) ~ (17) ? (18) " (19) . (20) (. IS NORMALLY NUMERIC)
, (21)
```

```
|KEYEX| TELLS NUMBER OF THE LARGEST TOKEN TO KEEP
OTHERS SERVE ONLY AS DELIMITERS
```

```
IF KEYEX < 0 THEN NUMERIC FIELDS ARE NOT DECODED BUT ARE
TREATED AS ALPHAMERIC AND TOKEN IS SET TO 1 FOR THEM.
IN THIS CASE, ALPHABETICS ARE ALLOWED IN THE NUMERIC FIELD
(I.E. 48CA IS VALID) AND +, -, AND . ARE TREATED AS TOKENS
INSTEAD OF PARTS OF NUMERICS.
```

```
STOP IS SET TO THE CHARACTER THAT STOPPED THE SCAN -- I.E.
THE FIRST CHARACTER TO BE CONSIDERED ON THE NEXT CALL TO
```

MSCAN. NOTE THAT FOR ITOKEN > 0 THIS IMPLIES THAT STOP IS
THE CHARACTER AFTER THE TOKEN.

BLANKS ARE DELIMITERS AND NOT MADE INTO TOKENS
' (APOSTROPHIES) DELIMIT LITERAL DATA, \$ IS COMMENT DELIMITER

5/7/75 - EXTENSIVE CHANGES FOR PTOLEMY - S. PIEPER
8/26/76 - MUCH REWRITE FOR CDC/IBM VERSION.

IMPLICIT
INPBUF

LOGICAL*1 STOP

LOGICAL*1 MESS(1)
LOGICAL*1 DOLLAR /'\$'//, BLANK /' '/, DOT /'.'//
LOGICAL*1 STAR /'*'//

LOGICAL*1 NUM0 /CHARDATAVALUE<0>//, NUM9 /CHARDATAVALUE<9>//,
1 LETTRA /CHARDATAVALUE<A>//, LETTRZ /CHARDATAVALUE<Z>//
LOGICAL*1 LETTRE /'E'//, LETTRD /'D'//

WORDINIT

LOGICAL*1 CHARS(21), DELIMS(4)

NOW

DATA CHARS / '%', '#', '&', ':', ';',
1 '-', '+', '/', '*', '(',
2 ')', '=', '@', '<', '>',
3 '|', '|', '?', '"', '.',
4 ', ' /
DATA DELIMS / '+', '-', ' ', '\$' /

PRESETS

INCHST = INCH
SIGN = +1
VAL = 0
ICNT = 0
INCH = INCH - 1
IMESS=0
ICOM=0
IDEC = 0

PHASES
1 NOTHING COOKING

```

C      2 ALPHABETIC FIELD BEING PROCESSED
C      3 NUMERICAL FIELD BEING PROCESSED
C
C      INPHAS = 1
C
C      GET NEXT CHAR IF COMMENT SKIP IF NOT FIND TYPE
C
100  INCH=INCH+1
      IF (INCH.GT.NOCH) GO TO 710
      ICH=IBUF(INCH)
C      IS A LITERAL COMMING IN
      IF (IMESS.GT.0) GO TO 149
C      IS A COMMENT COMMING IN (IF SO SKIP UNTIL $ IS MET)
C
      IF (ICOM.EQ.0) GO TO 101
      IF (ICH.NE.DOLLAR) GO TO 100
      ICOM=0
      GO TO 100
C
C      BLANK0
C
101  IF ( ICH .EQ. BLANK ) GO TO ( 100, 210, 340 ), INPHAS
C
      ICHVAL = CHARACTERVALUE(ICH)
C
      NUMBER 0
C
      IF ( ICHVAL .GE. NUM0 .AND. ICHVAL .LE. NUM9 ) GO TO 300
C
      ALPHABETIC 0
C
      IF ( ( ICHVAL .GE. LETTRA .AND. ICHVAL .LE. LETTRZ ) .OR.
1    ICH .EQ. STAR ) GO TO 200
C
      IT IS A SPECIAL CHARACTER
C
      IS IT A PERIOD
C
102  IF (ICH.EQ.DOT) GO TO 320
C
      ALL OTHER SPECIAL CHARACTERS STOP A FIELD THAT IS IN PROGRESS
C
      GO TO (105, 210, 340), INPHAS
C
      FIRST CHECK FOR + - ' $
C
105  DO 129 I = 1, 4
      IF ( ICH .EQ. DELIMS(I) ) GO TO 140
129  CONTINUE
C
      NOW LOOK FOR TOKENS
C
      DO 139 ITOKEN = 1, 21

```

```

        IF ( ICH .EQ. CHARS(ITOKEN) ) GO TO 590
139  CONTINUE
    GO TO 9000
C
140  GO TO ( 141, 142, 143, 144 ), I
141  SIGN = +1
    ITOKEN = 7
    IF ( KEYEX .LT. 0 ) GO TO 590
    GO TO 302
142  SIGN = -1
    ITOKEN = 6
    IF ( KEYEX .LT. 0 ) GO TO 590
    GO TO 302
143  IMESS = 1
    GO TO 100
144  ICOM = 1
    GO TO 100
C
C
C  COLLECTING A LITERAL
C
149  MESS(IMESS)=ICH
    IMESS=IMESS+1
    IF(IMESS.GT.80) GO TO 9006
    IF (ICH.NE.DELIMS(3))GO TO 100
    IMESS=IMESS-2
    MESS(IMESS+1)=BLANK
    ITOKEN=-4
    GO TO 602
C
C
C  ALPHABETIC CHARACTER CHECK PHASE
C
C
200  GO TO(201,202,330),INPHAS
C  START
201  WORDSTART
202  WORDADD(ICH)
    INPHAS=2
    GO TO 100
C  DONE
210  IMESS = IWORD
    WORDSAVE(VALU)
    ITOKEN=-1
    GO TO 600
C
C
C  NUMERIC
C
300  IF ( KEYEX .LT. 0 ) GO TO 200
    GO TO (301,202,303),INPHAS
C  START
301  IDEC=0
    VAL = ICHVAL - NUM0
    ICNT = 1
302  POW=1.0

```



```

INPHAS=3
GO TO 100
C MORE
303 VAL = 10*VAL + ICHVAL-NUM0
IF (IDEC.NE.0) POW=POW*10.0
ICNT = ICNT + 1
GO TO 100

C
C DECIMAL PT.
C
320 ITOKEN = 20
IF ( KEYEX .LT. 0 ) GO TO 590
GO TO (321,9002,310),INPHAS

C
C DECIMAL PT ENCOUNTERED IN NUMERIC FIELD
C
310 IF (IDEC.NE.0) GO TO 9002
IDEC=1
GO TO 100
321 IDEC=1
VAL=0.0
GO TO 302

C
C ALPHA CHECK E AND D
C
330 CONTINUE
IF (ICH.EQ.LETTRE .OR. ICH .EQ. LETTRD ) GO TO 332
C
C
C GO TO 9012
C
C
C E OR D FORMAT
C
332 ISG = 1
IDEC = 1
IPOW=0
INCH=INCH+1
IF (INCH.GT.NOCH) GO TO 9008
ICH=IBUF(INCH)
IF (ICH.EQ.DELIMS(1)) GO TO 334
IF (ICH.NE.DELIMS(2)) GO TO 335
ISG=-1
334 INCH=INCH+1
IF (INCH.GT.NOCH) GO TO 3360
ICH=IBUF(INCH)
335 ICHVAL = CHARACTERVALUE(ICH)
IF ( ICHVAL .LT. NUM0 .OR. ICHVAL .GT. NUM9 ) GO TO 336
IPOW=IPOW*10+ ICHVAL-NUM0
CIB IF (IPOW.GT.75) GO TO 9004
CDC IF ( IPOW .GT. 300 ) GO TO 9004
GO TO 334
C
3360 CONTINUE

```

```

336 IF (IPOW.EQ.0)GO TO 340
    P2=10.0D0**IPOW
    IF (IS G.LT.0) GO TO 337
    POW=POW/P2
    GO TO 340
337 POW=POW*P2
    GO TO 340
340 ITOKEN=-2
    VALU1=VAL/POW
    VALU=VALU1*SIGN
    IVALU = VALU
    IF ( IDEC .EQ. 0 ) ITOKEN = -5
    IF ( ICNT .EQ. 0 ) GO TO 9010
    GO TO 600

C
C     SPECIAL CHARACTER FOUND - SHOULD WE SKIP OVER ITO
C
590 IF (ITOKEN.GT. IABS(KEYEX) )GO TO 100
    INCH = INCH + 1

C
C     FIELD HAS BEEN FOUND AND PROCESSED
C
600 CONTINUE
602 STOP = IBUF (INCH)
    RETURN

C
C     END OF LINE ENCOUNTERED
C
710 CONTINUE
    GO TO (701,210,340) ,INPHAS
701 CONTINUE
    ITOKEN = -8
    RETURN

C
C     ERROR MESSAGES
C
9000 PRINT 9001, ICH, ICH
9001 FORMAT ( '0**** UNRECOGNIZABLE CHARACTER ''', A1, ''', 5X,
CIE      1 '(HEX IS ', Z2, ').') )
CDC      1 '(OCTAL IS ', O2, ').') )
    GO TO 9019
9002 PRINT 9003
9003 FORMAT ( '0**** INVALID DECIMAL POINT.' )
    GO TO 9019
9004 PRINT 9005
9005 FORMAT ('0**** INVALID OR TOO LARGE EXPONENT FIELD.')
    GO TO 9019
9006 PRINT 9007
9007 FORMAT ('0**** FIELD IS TOO LONG.' )
    GO TO 9019
9008 PRINT 9009
9009 FORMAT ('0**** E OR D MUST BE FOLLOWED BY NUMBER')
    GO TO 9019
9010 PRINT 9011

```

```

9011 FORMAT ( '0**** MISPLACED DECIMAL POINT OR PLUS OR MINUS SIGN.' )
      GO TO 9019
9012 PRINT 9013
9013 FORMAT ( '0**** INVALID ALPHABETIC IN NUMMERIC FIELD' )
C
9019 PRINT 9023, ( IBUF(I), I = INCHST, INCH )
9023 FORMAT ( ' FIELD UP TO THE ERROR IS: ', 80A1 )
      ITOKEN = -9
C
      RETURN
      END

```

MSCAN - IBM Fortran

The following is the IBM Fortran version of the second example. The comments at the beginning of the subroutine have been removed for brevity.

```
SUBROUTINE MSCAN ( KEYEX, ITOKEN, VALU, IVALU, MESS, IMESS, STOP
*)
```

```
IMPLICIT REAL*8 ( A-H, O-Z )
GENERIC
LOGICAL*1 IBUF(80)
COMMON /INPBUF/ NOCH, INCH, IBUF
```

```
LOGICAL*1 STOP
```

```
LOGICAL*1 MESS(1)
LOGICAL*1 DOLLAR / 1H$/ , BLANK / 1H / , DOT / 1H./
LOGICAL*1 STAR / 1H*/
```

```
LOGICAL*1 NUM0 /1H0/ , NUM9 /1H9/ , LETTRA /1HA/ , LETTRZ /1HZ/
LOGICAL*1 LETTRE / 1HE/ , LETTRD / 1HD/
```

```
REAL*8 WORD, BLANK8 / 1H /
LOGICAL*1 WORD1(8)
EQUIVALENCE ( WORD, WORD1(1) )
```

```
LOGICAL*1 CHARS(21), DELIMS(4)
```

```
DATA CHARS / '%', '#', '&', ':', ';',
1 '!', '+', '/', '*', '(',
2 ')', '=', '@', '<', '>',
3 '|', '-', '?', '"', '\',
4 ',', ' ' /
DATA DELIMS / '+', '-', '"', '$' /
```

```
PRESETS
```

```
INCHST = INCH
SIGN = +1
VAL = 0
ICNT = 0
INCH = INCH - 1
IMESS=0
ICOM=0
IDEC = 0
```

```
PHASES
1 NOTHING COOKING
```

```

C      2 ALPHABETIC FIELD BEING PROCESSED
C      3 NUMERICAL FIELD BEING PROCESSED
C
C      INPHAS = 1
C
C      GET NEXT CHAR IF COMMENT SKIP IF NOT FIND TYPE
C
100  INCH=INCH+1
      IF (INCH.GT.NOCH) GO TO 710
      ICH=IBUF(INCH)
C      IS A LITERAL COMMING IN
      IF(IMESS.GT.0)GO TO 149
C      IS A COMMENT COMMING IN (IF SO SKIP UNTIL $ IS MET)
C
      IF(ICOM.EQ.0)GO TO 101
      IF (ICH.NE.DOLLAR) GO TO 100
      ICOM=0
      GO TO 100
C
C      BLANK0
C
101  IF ( ICH .EQ. BLANK ) GO TO ( 100, 210, 340 ), INPHAS
C
      ICHVAL = ICH
C
      NUMBER 0
C
      IF ( ICHVAL .GE. NUM0 .AND. ICHVAL .LE. NUM9 ) GO TO 300
C
      ALPHABETIC 0
C
      IF ( ( ICHVAL .GE. LETTRA .AND. ICHVAL .LE. LETTRZ ) .OR. ICH .EQ.
* STAR ) GO TO 200
C
      IT IS A SPECIAL CHARACTER
C
      IS IT A PERIOD
C
102  IF (ICH.EQ.DOT) GO TO 320
C
      ALL OTHER SPECIAL CHARACTERS STOP A FIELD THAT IS IN PROGRESS
C
      GO TO(105,210,340),INPHAS
C
      FIRST CHECK FOR + - ' $
C
105  DO 129 I = 1, 4
      IF ( ICH .EQ. DELIMS(I) ) GO TO 140
129  CONTINUE
C
      NOW LOOK FOR TOKENS
C
      DO 139 ITOKEN = 1, 21

```



```

        IF ( ICH .EQ. CHARS(ITOKEN) ) GO TO 590
139  CONTINUE
        GO TO 9000
C
140  GO TO ( 141, 142, 143, 144 ), I
141  SIGN = +1
        ITOKEN = 7
        IF ( KEYEX .LT. 0 ) GO TO 590
        GO TO 302
142  SIGN = -1
        ITOKEN = 6
        IF ( KEYEX .LT. 0 ) GO TO 590
        GO TO 302
143  IMESS = 1
        GO TO 100
144  ICOM = 1
        GO TO 100
C
C      COLLECTING A LITERAL
C
149  MESS(IMESS)=ICH
        IMESS=IMESS+1
        IF(IMESS.GT.80) GO TO 9006
        IF (ICH.NE.DELIMS(3))GO TO 100
        IMESS=IMESS-2
        MESS(IMESS+1)=BLANK
        ITOKEN=-4
        GO TO 602
C
C      ALPHABETIC CHARACTER CHECK PHASE
C
C
200  GO TO(201,202,330),INPHAS
C      START
201  WORD = BLANK8
        IWORD = 0
202  IF (IWORD .LT. 8) WORD1(IWORD+1) = ICH
        IF (IWORD .LT. 8) IWORD = IWORD+1
        INPHAS=2
        GO TO 100
C      DONE
210  IMESS = IWORD
        VALU = WORD
        ITOKEN=-1
        GO TO 600
C
C
C      NUMERIC
C
300  IF ( KEYEX .LT. 0 ) GO TO 200
        GO TO (301,202,303),INPHAS
C      START
301  IDEC=0
        VAL = ICHVAL - NUM0

```

```

        ICNT = 1
302    POW=1.0
        INPHAS=3
        GO TO 100
C
C    MORE
303    VAL = 10*VAL + ICHVAL-NUM0
        IF (IDEC.NE.0) POW=POW*10.0
        ICNT = ICNT + 1
        GO TO 100
C
C    DECIMAL PT.
C
320    ITOKEN = 20
        IF ( KEYEX .LT. 0 ) GO TO 590
        GO TO (321,9002,310),INPHAS
C
C    DECIMAL PT ENCOUNTERED IN NUMERIC FIELD
C
310    IF (IDEC.NE.0) GO TO 9002
        IDEC=1
        GO TO 100
321    IDEC=1
        VAL=0.0
        GO TO 302
C
C    ALPHA CHECK E AND D
C
330    CONTINUE
        IF (ICH.EQ.LETTRE .OR. ICH .EQ. LETTRD ) GO TO 332
C
C
        GO TO 9012
C
C
C    E OR D FORMAT
C
332    ISG = 1
        IDEC = 1
        IPOW=0
        INCH=INCH+1
        IF (INCH.GT.NOCH) GO TO 9008
        ICH=IBUF(INCH)
        IF (ICH.EQ.DELIMS(1)) GO TO 334
        IF (ICH.NE.DELIMS(2)) GO TO 335
        ISG=-1
334    INCH=INCH+1
        IF (INCH.GT.NOCH) GO TO 3360
        ICH=IBUF(INCH)
335    ICHVAL = ICH
        IF ( ICHVAL .LT. NUM0 .OR. ICHVAL .GT. NUM9 ) GO TO 336
        IPOW=IPOW*10+ ICHVAL-NUM0
        IF (IPOW.GT.75) GO TO 9004
CDC    IF ( IPOW .GT. 300 ) GO TO 9004
        GO TO 334

```

```

C
3360 CONTINUE
336 IF (IPOW.EQ.0) GO TO 340
    P2=10.0D0**IPOW
    IF (IS G.LT.0) GO TO 337
    POW=POW/P2
    GO TO 340
337 POW=POW*P2
    GO TO 340
340 ITOKEN=-2
    VALU1=VAL/POW
    VALU=VALU1*SIGN
    IVALU = VALU
    IF ( IDEC .EQ. 0 ) ITOKEN = -5
    IF ( ICNT .EQ. 0 ) GO TO 9010
    GO TO 600

C
C     SPECIAL CHARACTER FOUND - SHOULD WE SKIP OVER ITO
C
590 IF(ITOKEN.GT. IABS(KEYEX) )GO TO 100
    INCH = INCH + 1

C
C     FIELD HAS BEEN FOUND AND PROCESSED
C
600 CONTINUE
602 STOP = IBUF(INCH)
    RETURN

C
C     END OF LINE ENCOUNTERED
C
710 CONTINUE
    GO TO(701,210,340) ,INPHAS
701 CONTINUE
    ITOKEN = -8
    RETURN

C
C     ERROR MESSAGES
C
9000 PRINT 9001, ICH, ICH
9001 FORMAT (    32H0**** UNRECOGNIZABLE CHARACTER ', A1,      1H', 5X,
    *      8H(HEX IS , Z2,      2H). )
CDC   1  '(OCTAL IS ', O2, ').' )
    GO TO 9019
9002 PRINT 9003
9003 FORMAT (    28H0**** INVALID DECIMAL POINT. )
    GO TO 9019
9004 PRINT 9005
9005 FORMAT (    42H0**** INVALID OR TOO LARGE EXPONENT FIELD.)
    GO TO 9019
9006 PRINT 9007
9007 FORMAT (    24H0**** FIELD IS TOO LONG. )
    GO TO 9019
9008 PRINT 9009
9009 FORMAT (    39H0**** E OR D MUST BE FOLLOWED BY NUMBER)

```

```

      GO TO 9019
9010 PRINT 9011
9011 FORMAT (      52H0**** MISPLACED DECIMAL POINT OR PLUS OR MINUS SIGN
      *. )
      GO TO 9019
9012 PRINT 9013
9013 FORMAT (      42H0**** INVALID ALPHABETIC IN NUMMERIC FIELD )
C
9019 PRINT 9023, ( IBUF(I), I = INCHST, INCH )
9023 FORMAT (      27H FIELD UP TO THE ERROR IS: , 80A1 )
      ITOKEN = -9
C
      RETURN
      END

```

MSCAN - CDC Fortran

The following is the CDC Fortran version of the second example.

```

SUBROUTINE MSCAN ( KEYEX, ITOKEN, VALU, IVALU, MESS, IMESS, STOP
*)
C
C
  INTEGER IBUF(80)
  COMMON /INPBUF/ NOCH, INCH, IBUF
C
  INTEGER STOP
C
  INTEGER MESS(1)
  INTEGER DOLLAR
  INTEGER BLANK
  INTEGER DOT
  INTEGER STAR
C
  INTEGER NUM0
  INTEGER NUM9
  INTEGER LETTRA
  INTEGER LETTRZ
  INTEGER LETTRE
  INTEGER LETTRD
C
  REAL WORD
  REAL BLANK8
C
  INTEGER CHARS(21), DELIMS(4)
C
  DATA DOLLAR /      1H$/
  DATA BLANK /      1H /
  DATA DOT /      1H./
  DATA STAR /      1H*/
  DATA NUM0 /1R0/
  DATA NUM9 /1R9/
  DATA LETTRA /1RA/
  DATA LETTRZ /1RZ/
  DATA LETTRE /      1HE/
  DATA LETTRD /      1HD/
  DATA BLANK8 /      1H /
C
  DATA CHARS / '%', '#', '&', ':', ';',
1  '-', '+', '/', '*', '(',
2  ')', '=', '@', '<', '>',
3  '|', '~', '?', '"', '.',
4  ',', ' ' /
  DATA DELIMS / '+', '-', '!', '$' /

```

```

C   PRESETS
C
C   INCHST = INCH
C   SIGN = +1
C   VAL = 0
C   ICNT = 0
C   INCH = INCH - 1
C   IMESS=0
C   ICOM=0
C   IDEC = 0
C
C   PHASES
C   1 NOTHING COOKING
C   2 ALPHABETIC FIELD BEING PROCESSED
C   3 NUMERICAL FIELD BEING PROCESSED
C
C   INPHAS = 1
C
C   GET NEXT CHAR IF COMMENT SKIP IF NOT FIND TYPE
C
C   100 INCH=INCH+1
C       IF (INCH.GT.NOCH) GO TO 710
C       ICH=IBUF(INCH)
C       IS A LITERAL COMMING IN
C       IF(IMESS.GT.0)GO TO 149
C       IS A COMMENT COMMING IN (IF SO SKIP UNTIL $ IS MET)
C
C       IF(ICOM.EQ.0)GO TO 101
C       IF (ICH.NE.DOLLAR) GO TO 100
C       ICOM=0
C       GO TO 100
C
C   BLANK0
C
C   101 IF ( ICH .EQ. BLANK ) GO TO ( 100, 210, 340 ) , INPHAS
C
C       ICHVAL = (SHIFT(ICH,6) .AND. 77B)
C
C       NUMBER 0
C
C       IF ( ICHVAL .GE. NUM0 .AND. ICHVAL .LE. NUM9 ) GO TO 300
C
C       ALPHABETIC 0
C
C       IF ( ( ICHVAL .GE. LETTRA .AND. ICHVAL .LE. LETTRZ) .OR. ICH .EQ.
C * STAR ) GO TO 200
C
C       IT IS A SPECIAL CHARACTER
C
C       IS IT A PERIOD
C
C   102 IF (ICH.EQ.DOT) GO TO 320
C

```

```

C      ALL OTHER SPECIAL CHARACTERS STOP A FIELD THAT IS IN PROGRESS
C
C      GO TO(105,210,340),INPHAS
C
C      FIRST CHECK FOR + - ' $
C
105    DO 129 I = 1, 4
      IF ( ICH .EQ. DELIMS(I) ) GO TO 140
129    CONTINUE
C
C      NOW LOOK FOR TOKENS
C
      DO 139 ITOKEN = 1, 21
      IF ( ICH .EQ. CHARS(ITOKEN) ) GO TO 590
139    CONTINUE
      GO TO 9000
C
140    GO TO ( 141, 142, 143, 144 ), I
141    SIGN = +1
      ITOKEN = 7
      IF ( KEYEX .LT. 0 ) GO TO 590
      GO TO 302
142    SIGN = -1
      ITOKEN = 6
      IF ( KEYEX .LT. 0 ) GO TO 590
      GO TO 302
143    IMESS = 1
      GO TO 100
144    ICOM = 1
      GO TO 100
C
C      COLLECTING A LITERAL
C
149    MESS(IMESS)=ICH
      IMESS=IMESS+1
      IF(IMESS.GT.80) GO TO 9006
      IF (ICH.NE.DELIMS(3))GO TO 100
      IMESS=IMESS-2
      MESS(IMESS+1)=BLANK
      ITOKEN=-4
      GO TO 602
C
C      ALPHABETIC CHARACTER CHECK PHASE
C
C
200    GO TO(201,202,330),INPHAS
C      START
201    WORD = BLANK8
      IWORD = 0
202    IF (IWORD .LT. 8) WORD = (SHIFT(WORD,6) .AND. 77777777777777777700
      *B) .OR. (SHIFT(ICH,6) .AND. 77B)
      IF (IWORD .LT. 8) IWORD = IWORD+1
      INPHAS=2
      GO TO 100

```

```

C      DONE
210    IMESS = IWORD
      IWORD = 10-IWORD
      IF (IWORD .GT. 0) WORD = SHIFT (WORD, 6*IWORD)
      VALU = WORD
      ITOKEN=-1
      GO TO 600

C
C
C      NUMERIC
C
300    IF ( KEYEX .LT. 0 ) GO TO 200
      GO TO (301,202,303),INPHAS
C      START
301    IDEC=0
      VAL = ICHVAL - NUMO
      ICNT = 1
302    POW=1.0
      INPHAS=3
      GO TO 100
C      MORE
303    VAL = 10*VAL + ICHVAL-NUMO
      IF (IDEC.NE.0) POW=POW*10.0
      ICNT = ICNT + 1
      GO TO 100

C
C      DECIMAL PT.
C
320    ITOKEN = 20
      IF ( KEYEX .LT. 0 ) GO TO 590
      GO TO (321,9002,310),INPHAS

C
C      DECIMAL PT ENCOUNTERED IN NUMERIC FIELD
C
310    IF (IDEC.NE.0) GO TO 9002
      IDEC=1
      GO TO 100
321    IDEC=1
      VAL=0.0
      GO TO 302

C
C      ALPHA CHECK E AND D
C
330    CONTINUE
      IF (ICH.EQ.LETTRE .OR. ICH .EQ. LETTRD ) GO TO 332
C
C
      GO TO 9012

C
C
C      E OR D FORMAT
C
332    ISG = 1
      IDEC = 1

```



```

      IPOW=0
      INCH=INCH+1
      IF (INCH.GT.NOCH) GO TO 9008
      ICH=IBUF(INCH)
      IF (ICH.EQ.DELIMS(1)) GO TO 334
      IF (ICH.NE.DELIMS(2)) GO TO 335
      ISG=-1
334  INCH=INCH+1
      IF (INCH.GT.NOCH) GO TO 3360
      ICH=IBUF(INCH)
335  ICHVAL = (SHIFT(ICH,6) .AND. 77B)
      IF ( ICHVAL .LT. NUM0 .OR. ICHVAL .GT. NUM9 ) GO TO 336
      IPOW=IPOW*10+ ICHVAL-NUM0
CIB   IF (IPOW.GT.75) GO TO 9004
      IF ( IPOW .GT. 300 ) GO TO 9004
      GO TO 334
C
3360 CONTINUE
336  IF (IPOW.EQ.0) GO TO 340
      P2=10.0E0**IPOW
      IF (IS G.LT.0) GO TO 337
      POW=POW/P2
      GO TO 340
337  POW=POW*P2
      GO TO 340
340  ITOKEN=-2
      VALU1=VAL/POW
      VALU=VALU1*SIGN
      IVALU = VALU
      IF ( IDEC .EQ. 0 ) ITOKEN = -5
      IF ( ICNT .EQ. 0 ) GO TO 9010
      GO TO 600
C
C   SPECIAL CHARACTER FOUND - SHOULD WE SKIP OVER ITO
C
590  IF(ITOKEN.GT. IABS(KEYEX) )GO TO 100
      INCH = INCH + 1
C
C   FIELD HAS BEEN FOUND AND PROCESSED
C
600  CONTINUE
602  STOP = IBUF(INCH)
      RETURN
C
C   END OF LINE ENCOUNTERED
C
710  CONTINUE
      GO TO(701,210,340) ,INPHAS
701  CONTINUE
      ITOKEN = -8
      RETURN
C
C   ERROR MESSAGES
C

```

```

9000 PRINT 9001, ICH, ICH
9001 FORMAT ( 32H0**** UNRECOGNIZABLE CHARACTER ', A1, 1H', 5X,
* 10H(OCTAL IS , O2, 2H). )
GO TO 9019
9002 PRINT 9003
9003 FORMAT ( 28H0**** INVALID DECIMAL POINT. )
GO TO 9019
9004 PRINT 9005
9005 FORMAT ( 42H0**** INVALID OR TOO LARGE EXPONENT FIELD.)
GO TO 9019
9006 PRINT 9007
9007 FORMAT ( 24H0**** FIELD IS TOO LONG. )
GO TO 9019
9008 PRINT 9009
9009 FORMAT ( 39H0**** E OR D MUST BE FOLLOWED BY NUMBER)
GO TO 9019
9010 PRINT 9011
9011 FORMAT ( 52H0**** MISPLACED DECIMAL POINT OR PLUS OR MINUS SIGN
*. )
GO TO 9019
9012 PRINT 9013
9013 FORMAT ( 42H0**** INVALID ALPHABETIC IN NUMMERIC FIELD )
C
9019 PRINT 9023, ( IBUF(I), I = INCHST, INCH )
9023 FORMAT ( 27H FIELD UP TO THE ERROR IS: , 80A1 )
ITOKEN = -9
C
RETURN
END

```