MIW
[illegible]

UNIVERSITY COMPUTER CENTER NOTES & COMMENTS

Volume 4, Number 1
November, 1969

UNIVERSITY OF MINNESOTA
MINNEAPOLIS, MINNESOTA

TABLE OF CONTENTS

$$$

Beginning December 1, 1969, the following rates will be effective for non-student tabulating and keypunch jobs:

tabulating operator :	$4.25/hour
keypunch with operator:	4.00/hour
verifier with operator:	4.50/hour

The student rate for all jobs will remain unchanged.

$$$

********* THE UNIVERSITY OF MINNESOTA FORTRAN COMPILER *********

Control Card

The control card to obtain the compilation and execution of a Fortran source and data deck has the characters MNF followed by a period if the implied parameters of the MNF compiler are to be used or by an optional free-form parameter list. The characters MNF (pronounced NUF) stand for MiNnesota Fortran. Blank columns may be used in the optional parameter list for readability, but are ignored by the MNF compiler.

Examples:

```
MNF.
MNF(BCØ)
MNF(R=0,I=TAPE1,L=TAPE2,P=999,E=2)
```

The standard control card MNF. is equivalent to the following control card

```
MNF(I=INPUT,LR=OUTPUT,E=0,P=4096)
```

i.e.
1) Fortran Source from the input card deck
2) Source and Cross-Reference listing to the printer
3) All levels of error messages
4) Line limit on execution of 4096 lines

If any errors are found on the MNF control card the message
"ILLEGAL CONTROL PARAMETER character"
where "character" is the illegal character on the control card will appear.

Parameter List

Character	Explanation and/or Action
B	BINARY FILE OF OBJECT ROUTINES TO APPEND

If B is ABSENT, the Fortran source deck is compiled to binary and all unsatisfied external requests are obtained if possible from the Fortran subroutine library.

If B is PRESENT, the Fortran source deck is compiled to binary, all binary routines on the file LGØ are loaded after the compiler's binary object code, and then all unsatisfied externals are obtained from the Fortran subroutine library. This allows Compass and other Fortran compilers to supply subprograms to the MNF compiler. The compiler OPENS and REWINDS the LGØ file before loading it

If B=FILENAME, the action is identical to B PRESENT, except that FILENAME rather than LGØ is the file of binary routines which are appended to the MNF compiler's generated binary object code.

Examples of control card sequences:

```
FTN.          FUN(S)
MNF(BC)       MNF(B)
```

C CALLING SEQUENCE FOR EXTERNAL ROUTINES

If C is ABSENT, the RUN-FUN or Chippewa compiler calling sequence is generated for subprograms.

If C is PRESENT, the FTN or FORTRAN EXTENDED calling sequence is generated for subprograms.

D DEBUGGING MODE

If D is ABSENT, COMMENT cards with C$ in the first two columns are considered normal COMMENT statements and FATAL-TO-EXECUTION errors insure that the compiler does not go into the execution phase.

If D is PRESENT, COMMENT source cards with C$ in the first two columns are treated as normal Fortran statements and the first two columns are ignored. This gives two options to the Fortran user:

1) Standard Fortran statements that are necessary for debugging a Fortran source deck may be left in the deck for possible future use. The only limitation is that the C$ in the first two columns leaves only three columns for statement numbers on these debugging statements.

2) Non-standard Fortran trace and listing statements of the MNF compiler will be invisible when the source deck is used on other Fortran compilers if these statements have C$ in the first two columns.

Examples of C$ Fortran statements:

```
C$    MAP
C$    PAGE
C$    TRACE SUBSCRIPTS
C$    PRINT 10,A,B,C
C$ 10 FORMAT(4H ABC,3G20.10)
```

Besides the C$ option, the D being present means that the compiler will enter the execution phase even though there are FATAL-TO-EXECUTION errors. The object code will then run until it reaches the ABORT requests inserted by the FATAL-TO-EXECUTION errors.

E ERROR LEVEL OF DIAGNOSTICS

If E is ABSENT, all five levels of error messages (NON-USASI, CAUTION, WARNING, FATAL-TO-EXECUTION, and DEADLY-TO-COMPILATION) will go to the file OUTPUT (see OUTPUT) after the source statement causing such a message.

If E=decimal digit, all error messages of that level and lower are not output. Each level of message is assigned a digit value. The levels are:

1= NON-USASI
2= CAUTION
3= WARNING
4= FATAL-TO-EXECUTION

The level, DEADLY, is not assigned a value since only two messages may occur at this level: MACHINE ERROR and NOT ENOUGH STORAGE FOR COMPILATION. When the latter message occurs, all current generated binary code is cleared and the compiler continues compiling the remaining Fortran statements. Those programmers who use octal constants, multiple statements per card, multiple replacement, and expressions in the DO parameter and output lists will quickly tire of the many NON-USASI messages and use E=1 as a compiler parameter. Since the MNF compiler tries to identify all errors or possible errors, it will sometimes give out CAUTION messages for correct code. In this case, E=2 is useful in avoiding the CAUTION messages.

I INPUT FILE FOR FORTRAN SOURCE

If I is ABSENT, the Fortran source statements will be taken from the file INPUT (i.e., the same one the control cards are on). This is also equivalent to I=INPUT as a parameter.

If I is PRESENT, the Fortran source statements to compile will be taken from the file INPUT.

If I=FILENAME, the Fortran source statements to compile will be taken from FILENAME.

L LISTING OF THE FORTRAN SOURCE STATEMENTS

If L is ABSENT, the Fortran source listing is put out on the file OUTPUT (see OUTPUT). This is equivalent to L=OUTPUT.

If L is PRESENT, the Fortran source listing is put out on the file OUTPUT (see OUTPUT).

If L=0, the Fortran source listing is not put out.

If L=FILENAME, the Fortran source listing is put out on the file FILENAME, and the OUTPUT file (see OUTPUT) is changed to FILENAME.

The Fortran source listing may be further manipulated by the Fortran statements PAGE, LIST, and NOLIST, which may appear between Fortran subprograms or individual Fortran statements and thus give control over the spacing of source on the output and control as to which source statements may appear on the output. Note that a NOLIST statement will also suppress the Ø and the R output options until the next LIST statement.

Ø OBJECT CODE LISTING

If Ø is ABSENT, the mnemonic pseudo-Compass listing is not put out on the OUTPUT file (see OUTPUT).

If Ø is PRESENT, the mnemonic pseudo-Compass listing for each Fortran statement is put out after that Fortran statement on the file OUTPUT, if the L and LIST options are currently set.

If Ø=FILENAME, the mnemonic pseudo-Compass listing is put out on the file FILENAME and the OUTPUT file name (see OUTPUT) is changed to FILENAME.

If the Fortran source is being put out, the Fortran mnemonic pseudo-Compass listing may be further manipulated by the Fortran statements MAP and NOMAP which may appear between Fortran subprograms or individual Fortran statements.

Thus, these statements give control over which sections of the Fortran source will be followed by the object code listing. Note that a NOLIST will also suppress the object code listing.

OUTPUT — All four distinct output line images (Fortran source, pseudo-Compass, Errors, and Cross-Reference) go to the file OUTPUT or to that file which was last equated to one of the letters L, Ø, or R.

Thus, MNF(L=OUTPUT, R=TAPE1, Ø=TAPE2) will be equivalent to MNF(LØR=TAPE2) since only one disk buffer is supplied for these line images.

P — PAGE (currently LINE) COUNT LIMIT

If P is ABSENT, the execution line count limit is set to 4096 (approximately 64 pages of 64 lines).

If P is PRESENT, the execution line count is set negative, thus the first output will terminate the job.

If P=decimal number, the execution line count is set to that decimal number. At some future date this number will be a PAGE rather than a LINE count limit.

R — CROSS-REFERENCE LISTING

If R is ABSENT, the cross-reference listing is put out on the file OUTPUT (see OUTPUT).

If R is PRESENT, the same action occurs as R ABSENT.

If R=0, the cross-reference listing is not put out.

If R=FILENAME, the cross-reference listing is put out on file FILENAME and the OUTPUT file name (see OUTPUT) is changed to FILENAME.

The cross-reference listing may be further manipulated by the Fortran statements REFERENCES and NOREFERENCE which may appear between Fortran subprograms and individual Fortran statements. Thus, these statements give control over which sections of a Fortran source will provide names and numbers to the cross-reference listing. Note that a NOLIST will also suppress the cross-reference listing.

T TRACE POSSIBLE ERRORS

If T is ABSENT, normal object execution code is produced for Fortran statements.

If T is PRESENT, all of the standard invisible MNF RUNTIME TRACE routines are included in the execution code. By invisible we mean that only if an error occurs will a message be output which details Fortran source statement number, name of any variable involved and value that caused the message.

These are:

TRACE DO LOOPING - checks for initial greater than terminal, infinite or over 1000000 indexing.

TRACE FORMATIØ - checks type of the list variable vs. type of the format specification.

TRACE STATEMENT NUMBERS
TRACE SUBROUTINES
{ counts number of times each statement number is executed; also counts number of times each subroutine is executed; count put out at job finish.

TRACE SUBSCRIPTS - checks subscript to be between 1 and total dimension product. Checks actual arrays and dummy arrays for legality of dimension information.

TRACE TRANSFERS - checks assign variable against GØTØ number list and computed GØ TØ variable against total GØTØ list number.

Design of the Compiler

The compiler has two modes of operation, NORMAL and BATCH. In the NORMAL mode a Fortran source input record is passed against the compiler which resides in the lower portion of the field length. At the upper end of the field length are the input and output disk buffers. Source statements (INPUT file), previously compiled binary object decks (LGØ file), and Fortran subprogram library object decks are loaded through the input buffer. The output buffer receives the source object and cross-reference listings produced by the compiler.

The Fortran source statements are compiled to binary object code which starts next to the compiler and grows toward upper core. At the same time, dynamic temporary tables within a subprogram and permanent tables for the entire program grow from the I/O buffers in upper field length toward the binary object code. If the object code and tables collide, the compiler gives up. This is the only case (besides machine error) in which the compiler gives up compiling an executable program. To help avoid this collision and to obtain small field lengths for multiprogramming, the Fortran programmer should use the following techniques:

1) Blank COMMON is the only declarative code which does not get assigned immediate space in the object deck since it is assigned that location immediately after all programs, labeled COMMONs, and external routines. Thus, placing all non-initialized large arrays into COMMON will give extra compilation space.

2) The temporary tables are largest for large subprograms. Thus, placing these early in the source deck means that a larger total program may be compiled within a given space. It is also a good programming technique to keep subprograms small so that they are manageable and more easily checked out.

3) Putting R=0 will save core if only one large subprogram is used.

After the source Fortran is compiled into object code, the binary object decks of LGØ (if B is specified) are added to the binary already compiled. All remaining external requests are filled (if possible) from the normal Fortran library and linked together by the compiler's loader. The compiler then transfers to a short move program below cell 100_8 in the field length which moves the binary object code down on top of the compiler since all room from 100_8 to the field length (CM) is given over to the complete compiled binary object deck with a request to reduce core to the minimum needed by the object program.

Remember these hints for small compilation field length:

1) Large non-preset-by-data arrays put in blank COMMON.

2) Largest subprograms first in the source deck.

In the BATCH mode of operation, many (50-60) jobs consisting of JOBCARD, EØR, Fortran source, EØR, Data (if any), EØR, are passed against the compiler as in the NORMAL mode of operation. But, instead of moving the object code down on top of the compiler the control is passed to a subcontrol (PP routine) which reduces the subcontrol field length to that space that encompasses the binary object routine, i.e., that above the compiler code and below the I/O buffers of the compiler. Small buffers in the subcontrol are filled by and emptied to the large main disk buffers of the compiler. The subcontrol monitors time limit, error conditions, and RA+1 requests which are passed to the compiler. The main use of this is for student jobs in order that the jobs come out exactly in the same order as they went in for ease of matching input and output. It also reduces the requirements on the operating system's time and memory locations in that one FNT entry is the same for 60 jc and the overhead for starting up and terminating a job is 1/60th of that for the same number of jobs in NORMAL mode.

Current Report on the University of Minnesota MNF compiler

A pre-release version of the compiler will be added to the library system on December 1, 1969. It will be useful to detect program source compilation errors not detected by our other three Fortran compilers RUN, FUN, and FTN. The control card for MNF is described on page 5 of this newsletter. On Sunday, December 21, from 1 PM to 5 PM, there will be free runs of up to 5 minutes for users who wish to test their programs on the MNF compiler. Jim Mundstock and Lawrence Liddiard, the two main implementors and designers of the compiler, will be on site to answer questions regarding error messages, speed of compilation, and to bandage any errors that still might be in the compiler. The compilation rate will be poor on the pre-release compiler since the compiler sumchecks itself after every Fortran source statement to insure that it has not been changed (i.e., the compiler is re-entrant). The necessary core for the pre-release version is also higher than the final release (use about 60K octal) since it will still contain many compiler debugging aids.

Note the main differences between the other Fortran compilers and MNF are that MNF compiles directly to core (there is no LGØ file on the disk) and that MNF tries to detect as many compile and execution time errors as possible, while at the same time giving an excellent cross-reference listing. MNF will not replace the other compilers FUN and FTN since, in many cases, they compile better execution code, especially for DO loops. MNF produces good object code in the following areas:

a) Parameter substitution
b) Double and complex arithmetic
c) Register content remembrance
d) Fewer jump commands in arithmetic IF
e) Logical and relational expression evaluation
f) Smaller total core required for execution code

The current pre-release version of MNF has these defficiencies:

a) Object time trace routines are not finished
b) I/O uses current FUN techniques, thus requiring 2 words/I/O simple list element rather than the final version 1 word/I/O simple list element. Also the FORMAT FREE PRINT and READ are in the compiler but not MNF's CODED I/O routines.

TO ALL PROBLEM SPONSORS

The purpose of this newsletter is to disseminate information concerning the Computer Center to all persons using our facilities. In most cases, this is the only way in which we can inform programmers of changes or errors in the system, of new systems, and of changes in practices and policies.

With this in mind, we are asking that every problem sponsor see that the name of his problem manager (and the names of all programmers actually using the computer) be placed on our mailing list. Please send the names and addresses to:

Mrs. Amy Koepke
University Computer Center
227 Experimental Engineering

PLOTTING FACILITIES

There are two plotting options presently available to the programmer who generates plot tapes on the CDC 6600:

1) He may use the CDC 165 paper plotter. First, apply for a CDC 160 computer user's number at the office of the Hybrid Computer Laboratory in Room 142 Space Science Center. (This number is <u>not</u> the same as that assigned to 6600 projects.) After this, jobs may be submitted for plotting in Room 134 Space Science. The charge for the system is $35.00 per hour.

2) He may use the Calcomp 835 microfilm plotter. Jobs to be plotted on microfilm may be submitted at any UCC I/O room. As yet there is no charge for the system, but the cost for supplies will be charged against the user's UCC account number.

FROM THE WEST BANK

1) Schedule change:
In addition to the previously announced operating schedule, the West Bank Station will be connected to the 6600 on Saturday from 9 AM to 12 Noon (depending on the Lauderdale schedule).

2) CDC 3200 Library:

   a) Random Number Generators - Two Fortran callable pseudo-random number generators are available on the 3200. The use of these routines (IRAN and RAN2F), is based on 6600 documentation. For information, contact Hugh Jurgens in Room 93, Blegen Hall, (373-7875).

   b) Matrix Manipulation - MXLNEQ is available for the 3200 system. It is not part of the system, but the Compass Language card deck can be obtained from Hugh Jurgens. MXLNEQ follows the 6600 documentation.

c) Additions to the Library - Any user who has programs of possible general interest and who wishes to make these programs available to others may do so by contacting William Craig in Room 25 Blegen Hall (373-5599).

SORT/MERGE PACKAGE

In response to a letter from D.R. Lienke about the SORT/MERGE package, a CDC representative made the following observations which may be of interest to some of our users:

'Generally, any SORT is faster on disk. Disk is most efficiently used to sort short files but also long, well ordered files.

Tape sort is most efficiently used for long files; polyphase for long files not well ordered and balanced for long, well ordered files.

Sort efficiency is dependent on record length, field length, type of input file, type of disk, and number of tapes.

There is an error in the Reference Manual concerning Transfer to Owncode (page 3-2, paragraph 3.1)

Version 3.0, Sort/Merge Reference Manual will read:

> "Transfer to owncode routine is accomplished with an RJ instruction which fills the entry point of the owncode routine with an unconditional jump to the main routine."'

LIBRARY CHANGES & ADDITIONS

As of November 10, the following changes have been made:

BESJ	- new version
GAMMAF	- corrected error messages
LSQORPY	- corrected error in special case
MXLNEQ	- Compass version
MXMPLY1	- corrected error in special case
QLENGTH	- corrected 3.1.6 version
RCVECT	- correction in error message check
RESETFL	- corrected 3.1.6 version
UMST500	- Durbin-Watson D-statistic now calculated after back-solution; endfile required between problem data sets if separate data file used.
UMST550	- endfile required between problem data sets if separate data file used.
UMST560	- minor correction
UMST600	- if a frequency count is <u>not</u> selected, an unlimited number of observations and up to 999 variables may be specified as long as the field length requested is at least 50000.
UMST620	- Mode 3 now handles missing data correctly.
UMST630	- file handling corrections
ORTHON2	- new routine

BMD library changes as of November 15, 1969:

BMD02D	- file name correction
BMD03D	- 3.1.6 correction
BMD05D	- minor correction
BMD07D	- 3.1.6 correction
BMD01M	- 3.1.6 correction
BMD03M	- file name correction; 3.1.6 correction
BMD06M	- several corrections
BMD07M	- file name correction
BMD02R	- file name correction
BMD03R	- file name correction
BMD02S	- file name correction
BMD04S	- file name correction; 3.1.6 correction
BMD05S	- file name correction; 3.1.6 correction
BMD09S	- file name correction
BMD10S	- file name correction
BMD02T	- graph correction
BMD02V	- file name correction
BMD03V	- file name correction
BMD05V	- file name correction
BMD06V	- file name correction
BMD07V	- minor correction

CONSULTANT'S CORNER

Consider the following:

```
          IN = 1
          IT = 63
          DO 5 I = IN IT
        5 PRINT 2, I,I
        2 FORMAT(1X,Ø2,2X,R1)
          END
```

QUESTION: How many lines are printed by this program?
ANSWER : 1

The error illustrated here is a rather simple one, although it may not be obvious at first glance. The comma missing from the supposed DO statement results in a legal assignment statement. The value of the variable INIT is assigned to a variable named DO5I.

Admittedly, this is a somewhat specialized error (I have seen this in actual practice only once) but the effect of this type of error is not uncommon. Computation is being carried out with undefined quantitites. In the example given, INIT and I have not been defined, and DO5I has not been defined in a meaningful way. When a variable is undefined, there is no way in which to predict what value has been used for it.

Of course, there are many other ways in which undefined variables can occur. Simple spelling and keypunching errors are probably the most common offenders. A classic example is using NØ and NO to denote the same variable. Often, the programmer may simply forget to initialize something, or may commit subscripting errors, so that some of the elements of an array are neglected.

These errors, when embedded in longer programs, may be hard to find. I have seen programs which have given results based on undefined variables in which the error has gone undetected for many months.

Also, the programmer may not depend on the computer to recognize this situation. In the above example, only the FTN compiler will give an indication that something may be wrong with the statements.

STEVE LEGENHAUSEN

NOTE TO ALL USERS

All Users are hereby notified that any permission, either explicitly granted or implied for the storage of user supplies or furniture and other personal property in the Lauderdale Building has been revoked. Henceforth, no user property may be stored at Lauderdale without written permission from the Directory of the University Computer Center. User's property for which such permission has not been obtained and which is still in the building on December 12, 1969 will be removed by the University Computer Center.

Effective immediately the Lauderdale Building will not be open outside of scheduled running hours for the 6600 computer. This means that all persons will be obliged to leave the premises when the computer is not scheduled. There will be no exceptions without written permission from the Director of the University Computer Center.