NOAA Technical Memorandum ERL SEL-32

U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
Environmental Research Laboratories

FORTRAN Concordance Manual

R. GREGG MERRILL

Space
Environment
Laboratory
BOULDER,
COLORADO
April 1974

ENVIRONMENTAL RESEARCH LABORATORIES

SPACE ENVIRONMENT LABORATORY

IMPORTANT NOTICE

Technical Memoranda are used to insure prompt dissemination of special studies which, though of interest to the scientific community, may not be ready for formal publication. Since these papers may later be published in a modified form to include more recent information or research results, abstracting, citing, or reproducing this paper in the open literature is not encouraged. Contact the author for additional information on the subject matter discussed in this Memorandum.

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
Environmental Research Laboratories

NOAA Technical Memorandum ERL SEL-32

FORTRAN CONCORDANCE MANUAL

R. Gregg Merrill

Space Environment Laboratory
Boulder, Colorado
April 1974

CONTENTS

FORTRAN Concordance Manual

R. Gregg Merrill

The FORTRAN concordance program prepares a coded, alphabetic concordance of all the names, statement numbers, DO-loop indices, unit numbers, etc., in one or more programs and subprograms; the codes reveal the use of these elements in replacement statements and in all FORTRAN commands. The user may obtain a single concordance of a program and all its subroutines in order to obtain complete documentation of all variables in commons.

1. INTRODUCTION

FORTRAN programs often require far more documentation of the use of variable and subprogram names, statement numbers, DO-loops, etc., than compilers provide. For example, most FORTRAN compilers can make a list of variable names and statement numbers indexed to compiled locations, but these locations have only an indirect relationship to the FORTRAN statements, and the list gives no clues to the use of the names and numbers. Because the compiler makes separate lists for each program and subprogram, the user must refer to several lists to determine the use of variables in commons.

The concordance program provides complete documentation of any one or more programs and subprograms. If a program and its subprograms use commons, the user can make a single concordance that fully documents the use of all common variables.

The concordance program consists of two parts: a contextual parser (rather than syntactical), written in FORTRAN by the author; and a sorter, written in FORTRAN by L. David Lewis, NOAA, that incorporates the concordance output format.

The parser extracts from each FORTRAN statement all variable names, statement numbers, unit numbers, etc.; codes them according to their use in the statement; and indexes them with the card identifier in columns 73 - 80, or with an identifier generated by the parser from the user's specifications. The identifier of the first card of a statement that uses continuation cards indexes the entire statement.

If the FORTRAN statements lack identifiers, a NUMBER card specifies them and provides a listing of the identified statements and the option of punching a deck.

The parser ignores all numeric and Hollerith constants. It removes all imbedded blanks outside Hollerith specifications so that blanks do not appear in the concordance.

The contextual parser cannot detect syntactical errors, though it does object to unbalanced parentheses in certain contexts, and detects certain errors in FORTRAN commands; however, debugging is hardly its purpose.

The parser correctly parses CDC3800 FORTRAN, ANSI FORTRAN, and FORTRAN II, except for B-, D-, F-, and I-cards in FORTRAN II.

When the seven characters FORMAT(begin a statement, the parser defines them as a FORTRAN command.

2. GENERAL PARSE CODES

In the descriptions that follow, angular brackets enclose descriptive text: <NAME> designates any variable name; square brackets designate optional elements: [()] means that parentheses may or may not appear.

The parser codes names and statement numbers in arithmetic replacement statements as follows:

<NUMBER> SN	Statement number.
<NAME> =	Simple variable on left side of replacement.
<NAME>() =	Array name on left side of replacement.
<NAME>[()] = ,	Variable subscript on left side of replacement, or argument of arithmetic statement function definition.
<NAME>	Name without code designates value of variable used on right side of replacement.
<NAME>()	Subscripted variable or function name on right side of replacement.
<NAME> ,	Argument of a function, or name used in a subscript. More than one comma defines the depth of the argument. The parser ignores parentheses used solely for algebraic grouping or convenience.

3. FORTRAN COMMAND CODES

The following alphabetic list of FORTRAN command examples, with the parser's output for each, serves as an index to the coding of names and numbers used in the commands.

In each example, the ellipsis replaces spaces up to column 73, and the left slash (\) replaces the delimiter between the parse and the identifier (see section 4 for the concordance format).

ASSIGN

```
144   ASSIGN 10 TO KBIT . . . TEST080
144  SN\TEST080
10  ASSIGN\TEST080
KBIT  ASSIGN\TEST080
```

BACKFILE

```
      BACKFILE 10, IUN . . . DK03060
10  BACKFILE\DK03060
IUN  BACKFILE\DK03060
```

BACKSPACE

```
      BACKSPACE IUN . . . TEST123
IUN  BACKSPACE\TEST123
```

BANK

```
      BANK, (2), PLUS, /A/, /B/ . . . TEST070
2  BANK()\TEST070
PLUS  BANK\TEST070
/A/  BANK\TEST070
/B/  BANK\TEST070
```

BLOCK DATA

```
      BLOCK DATA ARRAY . . . DK05008
BLOCKDATA  \DK05008
ARRAY  BLOCKDATA\DK05008
```

BUFFER

```
      BUFFER IN (IFORM, 1) (X(FWA), X(LWA)) . . . TEST119
IFORM  BUFFINU\TEST119
1   BUFFINM\TEST119
X()   BUFFIN\TEST119
FWA   BUFFIN ,\TEST119
X()   BUFFIN\TEST119
LWA   BUFFIN ,\TEST119
```

U codes the unit and M codes the mode.

```
      BUFFER OUT (10, IMD) (A, B) . . . TEST121
10   BUFFOTU\TEST121
IMD   BUFFOTM\TEST121
A   BUFFOT\TEST121
B   BUFFOT\TEST121
```

CALL

```
200    CALL CARD(R(I,J)**N3, ALL 3, F) . . . TEST096
200   SN\TEST096
CARD()   CALL\TEST096
R()   CALL ,\TEST096
I   CALL , ,\TEST096
J   CALL , ,\TEST096
N3   CALL ,\TEST096
ALL3   CALL ,\TEST096
F   CALL ,\TEST096
```

COMMON, BLANK

```
      COMMON A, V(20), Z(10,10,10) . . . TEST064
COM   COM\TEST064
A   COM\TEST064
V()   COM\TEST064
Z()   COM\TEST064
```

COMMON, LABELED AND NUMBERED

```
      COMMON /A/A(10)/B/B(5,5)/C/C,X . . . TEST065
/A/   COM\TEST065
A()   /A/\TEST065
/B/   COM\TEST065
B()   /B/\TEST065
/C/   COM\TEST065
C   /C/\TEST065
X   /C/\TEST065
```

COMPLEX TYPE DECLARATION

```
      COMPLEX A, B(10,10) . . . DK01051
A   COMP\DK01051
B()   COMP\DK01051
```

COMPLEX FUNCTION

```
      COMPLEX FUNCTION XMAT(A,B) . . . DK03001
XMAT()   CFUNC\DK03001
A   CFUNC ,\DK03001
B   CFUNC ,\DK03001
```

CONTINUE

```
2     C O N T I N U E . . . DK04006
2   SN\DK04006
CONTINUE   \DK04006
```

DATA (CDC and ANSI)

```
      DATA((GIB(I),I=1,10)=1.,2.,8(4.32)),(V=1.),...DK01068
     A   (((AR(I,J),J=6,11),I=10,20)=66(256.E26))...DK01069
GIB()   DATA\DK01068
I   DATA ,\DK01068
I   DATA , =\DK01068
V   DATA\DK01068
AR()   DATA\DK01068
I   DATA ,\DK01068
J   DATA ,\DK01068
J   DATA , =\DK01068
I   DATA , =\DK01068
DATA   \DK01068
```

Note that the identifier of the first card of a continued statement indexes all elements of the statement. 3800 FORTRAN ignores the order of subscripts in an implicit DO-loop in a DATA list despite allowing the above form.

```
      DATA GIB,V/1.,2.,3.,7*4.32,1.,4.,22./,AR(22)...DK09005
     A  /66*256.E+26/ . . . DK09006
GIB  DATA\DK09005
V  DATA\DK09005
AR()  DATA\DK09005
DATA  \DK09005
```

DECODE

```
      DECODE (8, 1, LOC(6))TEMP, FORT, ARG . . .DK03028
8  DECC\DK03028
1  DECF\DK03028
LOC()  DECS\DK03028
TEMP  DEC\DK03028
FORT  DEC\DK03028
ARG  DEC\DK03028
```

C codes the character count, F the format, and S the starting point.

DIMENSION

```
      DIMENSION HERCULES (10,20), ENIGMA(4,100) . . . DK01061
HERCULES()  DIM\DK01061
ENIGMA()  DIM\DK01061
```

DO

```
      DO 230 KNDX = JMIN, JMAX, JINK . . . DK01086
230  DO\DK01086
KNDX  DO\DK01086
JMIN  DO1\DK01086
JMAX  DO2\DK01086
JINK  DO3\DK01086
```

DOUBLE PRECISION TYPE DECLARATION

```
      DOUBLE PRECISION TEEPEE . . . DK01053
TEEPEE  DOUB\DK01053
```

DOUBLE PRECISION FUNCTION

```
      DOUBLE PRECISION FUNCTION ACRE(A, B) . . . DK05001
ACRE()  DFUNC\DK05001
A   DFUNC ,\DK05001
B   DFUNC ,\DK05001
```

END

```
       E N D PROGRAM . . . DK01091
END   \DK01091
```

ENDFILE

```
       ENDFILE IOUT . . . DK03027
IOUT  ENDFILE\DK03027
```

ENCODE

```
      ENCODE (ILNG, NFORM, ART)LOC(1), LOC(20) . . . DK03029
ILNG   ENCC\DK03029
NFORM   ENCF\DK03029
ART   ENCS\DK03029
LOC()   ENC\DK03029
LOC()   ENC\DK03029
```

C codes the character count, F the format, and S the starting point.

ENTRY

```
       ENTRY CROW . . . DK03005
CROW   ENTRY\DK03005
```

EQUIVALENCE

```
      EQUIVALENCE (A(3), C(2)), (B, X, V(2)) . . . DK01067
A()   EQUI\DK01067
C()   EQUI\DK01067
B   EQUI\DK01067
X   EQUI\DK01067
V()   EQUI\DK01067
EQUIVALENCE   \DK01067
```

The concordance lists all identifiers of EQUIVALENCE statements under the entry EQUIVALENCE as well as coding each variable appearing in the declaration.

EXTERNAL

```
      EXTERNAL FORTFN, CRISIS . . . DK03006
FORTFN   EXT\DK03006
CRISIS   EXT\DK03006
```

FORMAT

```
20      FORMAT (15(E7.3)) . . . DK03008
FORMAT   \DK03008
20   FORMAT\DK03008
```

FUNCTION

```
      FUNCTION STAR(X, Y) . . . DK05002
STAR()   FUNC\DK05002
X   FUNC ,\DK05002
Y   FUNC ,\DK05002
```

GO TO

```
      GO TO 166 . . . DK01076
166   GOTO\DK01076

      GO TO ISWT, (10,20,30,40) . . . DK01077
ISWT   GOTO\DK01077
10   GOTO ,\DK01077
20   GOTO ,\DK01077
30   GOTO ,\DK01077
40   GOTO ,\DK01077
```

```
      GO TO (10,20),A*B-C . . . DK01081
10  GOTO ,\DK01081
20  GOTO ,\DK01081
A  GOTO\DK01081
B  GOTO\DK01081
C  GOTO\DK01081
```

The parser allows all other forms of GOTO provided by CDC3800 FORTRAN.

IF

```
      IF(A*B - C*SINF(X))10,10,20 . . . DK01083
A  IF ,\DK01083
B  IF ,\DK01083
C  IF ,\DK01083
SINF()  IF ,\DK01083
X  IF , ,\DK01083
10  IF\DK01083
10  IF\DK01083
20  IF\DK01083

      IF(XF(A-X(I,J)*C,N,I+1).GE.2.**6)L=3 $ I=I+N...DK01085
XF()  IF ,\DK01085
A  IF , ,\DK01085
X()  IF , ,\DK01085
I  IF , , ,\DK01085
J  IF , , ,\DK01085
C  IF , ,\DK01085
N  IF , ,\DK01085
I  IF , ,\DK01085
L  =\DK01085
I  =\DK01085
I  \DK01085
N  \DK01085

      IF(EOF,20)22,42 . . . DK03030
20  IFEOF\DK03030
22  IF\DK03030
42  IF\DK03030

      IF(IOCHECK,NIO)32,52 . . . DK03031
NIO  IFIOCHECK\DK03031
32  IF\DK03031
52  IF\DK03031
```

```
      IF(UNIT,IU(K))62,64,66,68 . . . DKØ3Ø32
IU()  IFUNIT\DKØ3Ø32
K  IFUNIT ,\DKØ3Ø32
62  IF\DKØ3Ø32
64  IF\DKØ3Ø32
66  IF\DKØ3Ø32
68  IF\DKØ3Ø32
```

INTEGER TYPE DECLARATION

```
       INTEGER QUID, PRO, QUO . . . DKØ1Ø57
QUID  INTE\DKØ1Ø57
PRO  INTE\DKØ1Ø57
QUO  INTE\DKØ1Ø57
```

INTEGER FUNCTION

```
       INTEGER FUNCTION QUERY (A, B) . . . DKØ5ØØ3
QUERY()  IFUNC\DKØ5ØØ3
A  IFUNC ,\DKØ5ØØ3
B  IFUNC ,\DKØ5ØØ3
```

LOGICAL TYPE DECLARATION

```
      LOGICAL L1, L2 . . . DKØ1Ø58
L1  LOGI\DKØ1Ø58
L2  LOGI\DKØ1Ø58
```

LOGICAL FUNCTION

```
      LOGICAL FUNCTION NAND(L1, L2) . . . DKØ5ØØ5
NAND  LFUNC\DKØ5ØØ5
L1  LFUNC ,\DKØ5ØØ5
L2  LFUNC ,\DKØ5ØØ5
```

PAUSE

```
      PAUSE 777 . . . DKØ1Ø88
PAUSE  \DKØ1Ø88
777  PAUSE\DKØ1Ø88
```

PRINT

```
      PRINT 20,Q(3),(R(I),I=1,L1,L2),N(I) . . . DK03009
20  PRINTF\DK03009
Q()  PRINT\DK03009
R()  PRINT\DK03009
I  PRINT ,\DK03009
I  PRINT , =\DK03009
L1  PRINT , DO\DK03009
L2  PRINT , DO\DK03009
N()  PRINT\DK03009
I  PRINT ,\DK03009
```

F codes the format.

```
      PRINT 20 . . . DK03044
20  PRINTF\DK03044

      PRINT, I, X . . . DK03056
I  PRINT\DK03056
X  PRINT\DK03056
```

PROGRAM

```
      PROGRAM PARTEST . . . DK01001
PARTEST  PROG\DK01001

      PROGRAM FORT (A, J, Z) . . . DK02003
FORT()  PROG\DK02003
A  PROG ,\DK02003
J  PROG ,\DK02003
Z  PROG ,\DK02003
```

PUNCH

```
      PUNCH 40, K, X . . . DK03011
40  PUNCHF\DK03011
K  PUNCH\DK03011
X  PUNCH\DK03011

      PUNCH 20 . . . DK03045
20  PUNCHF\DK03045

      PUNCH, X . . . DK03058
X  PUNCH\DK03058
```

READ

```
      READ 10, A . . . DK03016
10  READF\DK03016
A   READ\DK03016

      READ (50, 160) ARRAY . . . DK03017
50  READU\DK03017
160  READF\DK03017
ARRAY  READ\DK03017

      READ INPUT TAPE 50, 160, ARRAY . . . DK03018
50  READU\DK03018
160  READF\DK03018
ARRAY  READ\DK03018

      READ (25) (M2(I,J), J=1,5) . . . DK03019
25  READU\DK03019
M2()  READ\DK03019
I  READ ,\DK03019
J  READ ,\DK03019
J  READ , =\DK03019

446   READ TAPE 25, BAMX (I) . . . DK03020
446  SN\DK03020
25  READU\DK03020
BAMX() READ\DK03020
I  READ ,\DK03020
```

U codes the unit and F codes the format.
All the above READ commands may lack lists.

REAL TYPE DECLARATION

```
      REAL  E,  F(20)  . . . DK01055
E  REAL\DK01055
F()  REAL\DK01055
```

REAL FUNCTION

```
      REAL FUNCTION MODEL (X) . . . DK05006
MODEL  RFUNC\DK05006
X  RFUNC ,\DK05006
```

RELEASE

```
      RELEASE 49, IUN . . . DK03026
49  RELEASE\DK03026
IUN  RELEASE\DK03026
```

RETURN

```
      RETURN . . . DK02006
RETURN  \DK02006
```

REWIND

```
      REWIND 44 . . . DK03024
44  REWIND\DK03024
```

SKIPFILE

```
      SKIPFILE I, 20, J, 48 . . . DK03059
I  SKIPFILE\DK03059
20  SKIPFILE\DK03059
J  SKIPFILE\DK03059
48  SKIPFILE\DK03059
```

STOP

```
      STOP  174 . . . DK01090
STOP  \DK01090
174  STOP\DK01090
```

SUBROUTINE

```
      SUBROUTINE CARD (A, I) . . . DK02004
CARD()  SUBR\DK02004
A  SUBR ,\DK02004
I  SUBR ,\DK02004

      SUBROUTINE CLEAR . . . DK05007
CLEAR  SUBR\DK05007
```

TYPE

TYPE declarations (TYPE INTEGER, TYPE REAL, etc.) are parsed in exactly the same way as the corresponding declarations without the word TYPE.

WRITE

```
      WRITE(40,20)A, J, R(I) . . . DK0301 2
40  WRITEU\DK0301 2
20  WRITEF\DK0301 2
A  WRITE\DK0301 2
J  WRITE\DK0301 2
R()  WRITE\DK0301 2
I  WRITE ,\DK0301 2
```

Other WRITE commands correspond to those for READ.

4. THE CONCORDANCE

The sorter sorts the output of the parser and formats its own output so that each parsed element appears only once followed by the identifiers of all its occurrences. This output constitutes a concordance of all programs and subprograms parsed at the same time.

The following lines are taken from the 42-page concordance of a program with 13 subprograms that use six labeled commons among them. These examples truncate the list of identifiers following a parsed element to five from the original seven; continuation lines have an additional identifier at the start of the line. A concordance always begins with the names of all common blocks.

```
/ALGOR/  COM       ALGR003  DGLT009  IN34009  INTL008  MDEL011
         PROD009   STUB016  TETI009  WIND009  XMDL012
10  DO             ALGR084  DFIN060  MDEL095  STUB079
10  GOTO           IN34214  NI12138  OUTP130  PROD105  TETI221
10  GOTO ,         OUTP047
10  SN             ALGR088  DFIN063  IN34227  MDEL096  NI12151
         PROD112   STUB081  TETI224  WIND032
6000  FORMAT       OUTP053
6000  WRITEF       OUTP052  OUTP138A OUTP158A OUTP158B
AIRGLO  WRITE      OUTP148  OUTP151  OUTP154
AIRGLO()           IN34100  IN34132  IN34157  IN34185  MDEL 73
AIRGLO()  =        IN34081  INTL079  MDEL070  TETI173  TETI177
AIRGLO()  /ARRAY1/          DFIN009  DGLT010  IN34010  INTL009
         OUTP013   PROD010  STUB017  TETI010  WIND010  XMDL013
AIRGLO()  IF ,     OUTP333  OUTP333  OUTP333  OUTP333
IHRSTD             STUB048  STUB119B
IHRSTD  =          STUB119B STUB119B
IHRSTD  /DIVERS/   DFIN019  DGLT020  IN34020  INTL019  INTP015
         OUTP023   PROD020  READ014  STUB027  TETI020  WIND020
IHRSTD  IF ,       STUB119B STUB120B
IHRSTD  READ       READ073
IHRSTD  WRITE      OUTP072
ZWDZ               ALGR038  ALGR040  ALGR050  ALGR052  ALGR060
ZWDZ  =            ALGR029
```

5. OPERATION

5.1. SCOPE Cards (local system)

The concordance program for the CDC3800 is on a magnetic tape in the NOAA/ERL Computing Facility tape library at Boulder, CO. The requisite SCOPE cards are as follows, where the single quote stands for the 7/9 punch:

```
'JOB(10),<project number>,<name>,<run time>[,<comments>]
'DEMAND,40000B
'TAPE,49=RO,H-304
'EQUIP,49=RO,(PARSER)
'EQUIP,1=<FORTRAN input unit, usually 60>
'EQUIP,2=<concordance output unit, usually 61>
'LOADMAIN,49,<run time>,10000,8
NUMBER<identifier & count><initial count><punch option>
```

5.2. NUMBER Card(s)

The 'LOADMAIN card must be followed by a NUMBER card that has the letters NUMBER in columns 1 - 6 and one of the following formats in subsequent columns. If this card is missing, the parser prints

```
BAD INITIAL NUMBER CARD.      <card image>
```

and returns to the executive.

The FORTRAN deck may have NUMBER cards at any appropriate place in addition to the one that must follow 'LOADMAIN. For example, NUMBER cards may precede each subprogram declaration to guarantee distinct identifiers for each subprogram. They may also precede ENTRY statements and logical subdivisions as desired. A deck so provided with NUMBER cards, excluding the required first one, may be put on a magnetic tape and read from logical unit 1. If FORTRAN card images are on a tape that lacks NUMBER card images, the requisite first NUMBER card may be either NUMBERAUTO or NUMBER=, as described below.

Identifier and count format, columns 7 - 14. This field represents columns 73 - 80 of a FORTRAN card. The optional identifier consists of any characters except =, and the count format consists of consecutive equals signs that specify the position and maximum number of digits in the card count. Thus the identifier is everything but the count format. For example, TEST=== specifies that the FORTRAN statements be identified in columns 73 - 79 as TEST001 through TEST999; ID == starting in column 8 specifies identifiers ID 01

through ID 99 in columns 74 - 78; PROG===A specifies identifiers PROGØØ1A through PROG999A.

If column 7 of the NUMBER card contains = and the remainder of the field is blank, the count runs from ØØØØØØ1 through 9999999 in columns 73 through 79.

As soon as the card count overflows the count format, the count restarts at 1, so be sure to provide enough equals signs.

If columns 7 - 1Ø contain AUTO, the parser provides deck identifiers consisting of the letters DK followed by a two-digit deck number and a three-digit card count. Each END statement, except the last, increments the deck number and restarts the card count. This option thereby provides identifiers DKØ1ØØ1 through DKØ1999 for the first program or subprogram, and DK99ØØ1 through DK99999 for the 99th. The deck count restarts at 1 after 99.

If the FORTRAN input cards already contain distinct identifiers in columns 73 - 8Ø, the NUMBER card specifies these identifiers if it is blank beyond column 6.

When columns 7 - 14 are other than blank, the parser produces a paged listing of each input FORTRAN program and subprogram with the identifiers provided by the parser for use with the concordance. When these columns are blank, the parser prints each subprogram name after the first as a verification of programs parsed.

If columns 7 - 14 contain punches but lack a count field or AUTO, the parser issues a page eject, prints

INVALID NUMBER CARD. AUTO ASSUMED. <card image>

issues another page eject, and initiates the AUTO option. If this occurs immediately after an END card, the first identifier is DKØ2 rather than DKØ1.

If columns 7 - 14 are blank even though the input deck lacks identifiers, the parser issues a page eject, prints

NO IDENTIFICATION. AUTO ASSUMED. <card image>

issues another page eject, and continues with the AUTO option until the next NUMBER card or end of file.

Initial count, columns 15 - 22. If you desire an initial card number other than 1, enter the desired number in any consecutive columns in the field 15 - 22; make sure that the number either matches the format specified by the count format or is seven or fewer digits if only column 7 contains an equals sign. If the card count overflows the count field, the count restarts at 1 rather than at the count specified in this field. The initial count applies to the first AUTO deck only.

Punch option, column 23. If you desire decks punched with the identifiers generated by the parser, punch the letter P in column 23.

5.3. Multiple Concordances

If you want a separate concordance for each of several separate decks in a single run, insert an end-of-file card followed by 'LOADMAIN described above and an appropriate NUMBER card ahead of each deck after the first one.

6. ACKNOWLEDGMENTS

The concordance parser was written in SNOBOL4 and then translated into FORTRAN using the author's FORSNO Package. This work was supported by the Advanced Research Projects Agency under ARPA order 2019. Completion and maintenance of the concordance program was supported in part by NOAA/ERL Computer Services.