HP 3000 Computer Systems

FORTRAN/3000 to HP FORTRAN 77/3000

Conversion Guide

HEWLETT PACKARD
19447 PRUNERIDGE AVENUE, CUPERTINO, CA 95014

Part No. 5957-4690
E0185

Printed in U.S.A. 01/85

CONTENTS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

INTRODUCTION

SECTION 1

This conversion guide explains how to use the FORTRAN/3000 to HP FORTRAN 77/3000 conversion utility.

Several changes must be made to a FORTRAN/3000 program before it can be compiled by the HP FORTRAN 77/3000 compiler. The conversion utility is a program that automatically performs many of these changes. Those transformations that the conversion utility performs are described in Section 2 ("What the Conversion Utility Does"). The conversion utility cannot, however, perform all transformations; those changes that must be performed manually are described in Section 3 ("What the Conversion Utility Does Not Do").

The conversion utility consists of a program and three command files. Using the conversion utility is described in Section 4 ("Using the Conversion Utility"). The command files can be customized to your needs, or new command files created. This is described in Section 5 ("Customizing Command Files").

The conversion utility reads a FORTRAN/3000 source file, performs a number of transformations, and places the results in a new file, which can then be processed by the HP FORTRAN 77/3000 compiler. When the new file is compiled, those constructs that differ in HP FORTRAN 77/3000 from FORTRAN/3000 but are not detected by the conversion utility cause the HP FORTRAN 77/3000 compiler to generate a warning or an error message. The source file must then be manually changed in those places. When the utility converts a statement, it flags it in two ways. It places a C in column 1 of the old statement, and right angle brackets (>) in columns 2 through 6. It places left angle brackets (<) in columns 72 through 80 of the converted statement. (You can modify the command file so that the angle brackets are not generated. Refer to Section 5.)

WHAT THE CONVERSION UTILITY DOES

SECTION 2

Several changes must be made to a FORTRAN/3000 program before it can be compiled by the HP FORTRAN 77/3000 compiler. Some of the transformations are performed automatically by the conversion utility. This section lists all of those transformations.

- The following table lists the FORTRAN/3000 directives that the conversion utility maps to the equivalent HP FORTRAN 77/3000 directives.

FORTRAN/3000	HP FORTRAN 77/3000
BOUNDS	RANGE
MAP	TABLES on
NOMAP	TABLES off
NOLIST	LIST off
LIST	LIST on
CODE	LIST_CODE on
NOCODE	LIST_CODE off
CHECK	CHECK_FORMAL_PARM
SEGMENT = <string>	SEGMENT '<string>'

  You can, if you wish, use the conversion utility to convert only compiler directives (and nothing else). How to do this is explained in Section 4.

- The FORTRAN/3000 logical operator .XOR. is replaced with .NEQV. (These operators are equivalent.)

- All ACCEPT statements are replaced with READ * statements.

- All DISPLAY statements are replaced with PRINT * statements.

- The PARAMETER statement in FORTRAN/3000 does not take parentheses; in HP FORTRAN 77/3000, the constant list of the PARAMETER statement must be enclosed in parentheses. The conversion utility inserts them as required.

- The logical constants of FORTRAN/3000 are equivalent to the octal constants of HP FORTRAN 77/3000. These logical constants are preceded by a percent sign and followed by an L, as, for example, %244L. The octal constants of HP FORTRAN 77 are followed by a B, as, for example, 244B. The conversion utility makes these changes.

- Character constants in octal representation in FORTRAN/3000 are preceded by a percent sign (%) and followed by a C, as, for example, %15C. The conversion utility changes these to the CHAR function, using octal representation for the number, as, for example, CHAR(15B).

- FORTRAN/3000 uses the .CC. command to check condition codes. In HP FORTRAN 77/3000, this becomes the CCODE() function.

- The syntax for character substrings in FORTRAN/3000 is:

  name[first:length]

  where *first* is an integer specifying the first character of the string and *length* is an integer specifying the number of characters of the string.

  The syntax in HP FORTRAN 77/3000 is:

  name(first:last)

  where *first* is an integer specifying the first character of the string and *last* is an integer specifying the last character of the string.

  The conversion utility replaces the brackets ("[ ]") of the former with parentheses ("( )"), and substitutes *length* with its actual calculation of the value for *last*. For example, the FORTRAN/3000 substring reference `userid[3:12]` is replaced with `userid(3:12+3-1)`.

- While free formatting in source programs is part of FORTRAN/3000, it is not permitted in HP FORTRAN 77/3000. You can use the conversion utility to change free-formatted source code to the standard FORTRAN 77 fixed-format conventions.

WHAT THE CONVERSION UTILITY DOES NOT DO

SECTION 3

The conversion utility cannot perform all transformations necessary to properly compile a FORTRAN/3000 program with the HP FORTRAN 77/3000 compiler. Many of the changes that must be performed manually are listed in this section. After processing a source file with the conversion utility, and then checking it for the constructs listed here, you can compile the program with the HP FORTRAN 77/3000 compiler, which should flag the remaining problems as errors and warnings.

- The only predefined files in HP FORTRAN 77/3000 are FTN05 ($STDIN) and FTN06 ($STDLIST). FORTRAN/3000 programs, on the other hand, can read from or write to a file referenced by other numbers. These units are established with file equations prior to running a program, which link a unit number to an existing file. For example, the file equation `FILE FTN02='filename'` permits a FORTRAN/3000 program to perform input/output on `filename`, referencing it as unit 2 (as, `READ (2...)`. If the FORTRAN/3000 source program reads from or writes to any file referenced as other than unit 5 or 6, the HP FORTRAN 77/3000 program must explicitly open it with an OPEN statement. You can do this two ways. One is to open the file directly, by specifying its name, as, for example, `OPEN(2,FILE='filename')`. If you want to use the same file equations set up for the FORTRAN/3000 program, you can open the file with the same name specified in the file equation, as, for example, `OPEN(2,FILE='FTN02')`. Either method now permits reads from or writes to unit 2.

- Programs sometimes depend on integer and logical data types being either specifically one or two words in length. FORTRAN/3000 programs that use the INTEGER and LOGICAL data types (without length specification) default to one word, while, in the absence of explicit typing (with the SHORT or LONG compiler directive), HP FORTRAN 77/3000 programs default to two words. In deciding which you want to use, if any, you should take into account the following and other factors: parameter passing problems that might occur when calling external procedures (including system intrinsics) that expect one or the other data type, the effects of equivalencing, the range of values being used, the amount of stack space being used, and the FORTRAN intrinsic functions being used. Where the particular type is critical, you should explicitly change the INTEGER type of FORTRAN/3000 to INTEGER*2. You can also use the SHORT or LONG compiler directive, or explicitly type all integers as either INTEGER*2 or INTEGER*4. Using the SHORT compiler directive in your HP FORTRAN 77/3000 program will ensure that all INTEGER and LOGICAL variables are the same size they were in the FORTRAN/3000 program.

- When calling procedures that pass INTEGER and LOGICAL parameters by *reference*, make sure that the actual and formal parameters have the same word length. (They should either both be single or both double integer.) If parameters passed by reference are not the same, the Segmenter issues an error message. When parameters passed by *value* do not both have the same word length, the compiler internally converts single integers to double.

- The following intrinsic functions are used for the double integer type in FORTRAN/3000. Since double integers are the default in HP FORTRAN 77/3000, these functions don't exist in HP FORTRAN 77/3000:

JABS	AJMAX0	JMIN1
IJINT	JMAX0	FLOATJ
JINT	JMAX1	JFIX
JDINT	AJMIN0	JSIGN
JMOD	JMIN0	JDIM

Use generic intrinsic functions for the above. You can create a customized command file to make these conversions. Refer to Section 5 for more information.

- The following FORTRAN/3000 functions don't exist in HP FORTRAN 77/3000. If your program uses one of these, the function should be replaced by an equivalent function, or some other way found to perform the task.

INUM	DNUM	CSINH
JNUM	STR	CCOSH
RNUM	BOOL	CTANH

- The syntax differs between the two compilers for calling a parameterless procedure, function, or intrinsic. Here is an example of how it is done in FORTRAN/3000:

```
time=clock
```

HP FORTRAN 77/3000 requires an empty pair of parentheses, as:

```
time=clock()
```

If your FORTRAN/3000 program contains many parameterless procedures of the same name you can create a customized command file to make the required changes. For details, refer to Section 5.

- Mixed mode expressions are evaluated differently in the two compilers. In operations of the same precedence, FORTRAN/3000 evaluates the same types within an expression first, while HP FORTRAN 77/3000 evaluates strictly from left to right. The following example program produces different results in the two compilers:

```
INTEGER*4 j
j=2000000000
WRITE(6,*) 1.0+j-j
END
```

The result returned in FORTRAN/3000 is 1.0
The result returned in HP FORTRAN 77/3000 is 0.0

You can use parentheses to force the order of evaluation you want. That is, if you want the HP FORTRAN 77/3000 program to give 1.0 as the answer, make this change:

```
WRITE(6,*) 1.0+(j-j)
```

- FORTRAN/3000's S edit descriptor for character data should be changed to HP FORTRAN 77/3000's A or R descriptor.

- FORTRAN/3000 uses partial word designators for bit extraction. These should be replaced with HP FORTRAN 77/3000's bit extraction intrinsics. For example, in FORTRAN/3000, `i[m:n]` refers to the subfield of *n* bits extracted from `i`, starting at position *m*, numbered from left to right, while `IBITS(i,a,b)` extracts the subfield of *b* bits from `i`, starting at position *a*, numbered from right to left.

  To change the *first* of the FORTRAN/3000 partial word designator to the second parameter of the IBITS function, use the following:

  ```
  16-(first bit + number of bits)
  ```

  For example, `i[0:7]` translates to `IBITS(i,(16-(0+7),7)`, or `IBITS(i,9,7)`. `i[9:4]` becomes `IBITS(i,3,4)`.

  Note that in FORTRAN/3000 substring designators and partial word designators have the same syntax. The difference is that a substring is recognized as being used with a variable typed earlier as CHARACTER while a partial word designator is used with a variable typed INTEGER or LOGICAL.

- In the PARAMETER statement in FORTRAN/3000, the type of a named constant is determined solely by the constant itself and not by the initial letter of its name. In HP FORTRAN 77, the type is determined by the initial letter of the name. For this reason, a variable should be explicitly typed before using the PARAMETER statement.

- The following FORTRAN/3000 compiler directives do not exist in HP FORTRAN 77/3000: CROSSREF, ERRORS, FILE, FIXED, INIT, LABEL, NOLABEL, LOCATION, STAT, NOSTAT, WARN, NOWARN, INTEGER*4, EDIT, TRACE, SOURCE, NOSOURCE.

USING THE CONVERSION UTILITY

SECTION 4

Use the command SHOWCATALOG to find out if the CONVERT UDC is in your catalog. You cannot use the conversion utility if it is not. To add the UDC to your catalog, execute this command:

```
:SETCATALOG FTNUDC.PUB.SYS[,otherudc]
```

where

otherudc is the list of any other UDC's that you are using.

The following UDC command runs the program FTNCVT.PUB.SYS on a FORTRAN/3000 source file.

```
:CONVERT oldsource[,[newsource][,commandfile]]
```

where

oldsource is the name of the FORTRAN/3000 source file to be converted to an HP FORTRAN 77/3000 source file.

newsource is the name of the HP FORTRAN 77/3000 file to be created. The default is $STDLIST. If *newsource* is not specified, the converted file is displayed to the terminal and no source file is saved.

commandfile is the name of the command file to be used. This is either one of the three command files supplied with the utility, and described below, or a command file of your own making. Creating your own command file is described in Section 5. If *commandfile* is not specified, the conversion operations supplied in the file FTNCMDS.PUB.SYS are used.

The three commands files included with the utility are:

- FTNCMDS.PUB.SYS performs all the conversions described in Section 2, except for operations on free format source.
- FTNFREE.PUB.SYS converts FORTRAN/3000 free format source to the fixed format required by HP FORTRAN 77/3000.
- FTNOPTN.PUB.SYS is a subset of the command file FTNCMDS.PUB.SYS. It operates only on compiler directives.

These command files can be modified, or new command files created to your own specifications. Refer to Section 5 for more information.

Here is an example that shows the steps in converting a FORTRAN/3000 source program to an HP FORTRAN 77/3000 source.

This is the program as originally written in FORTRAN/3000:

```
      PROGRAM time
      SYSTEM INTRINSIC DATELINE
      PARAMETER prompt = "The time is "
      CHARACTER datebuf*27
      CALL DATELINE(datebuf)
      DISPLAY prompt,datebuf[19:9]
      STOP
      END
```

This is the UDC command that processes the program by the conversion utility, producing an HP FORTRAN 77/3000 source called TIME77:

```
:CONVERT TIME,TIME77
```

While the conversion utility runs, it sends the following information to the terminal:

```
A.00.00 FTNCVT (C) HEWLETT-PACKARD CO. 1984

*
* convert source
*
* Convert Octal, Logical, and Character literals
* Convert substring indices
* Convert condition code access
* Convert console I/O (ACCEPT and DISPLAY)
* Convert .XOR. to .NEQV.
* Put parens around PARAMETER arguments (may still need declarations)
*
* compiler options
*
* convert "CHECK = n" to "CHECK_FORMAL_PARM n"
* convert "SEGMENT = string" to "SEGMENT  'string'"
* convert "LIST" to "LIST on"
* convert "NOLIST" to "LIST off"
* convert "NOCODE" to "LIST_CODE off"
* convert "CODE" to "LIST_CODE on"
* convert "NOMAP" to "TABLES off"
* convert "MAP" to "TABLES on"
* try to convert conditional compilation flags
```

Here is the converted file, TIME77:

```
      PROGRAM time
      SYSTEM INTRINSIC DATELINE
C>>>>>PARAMETER prompt = "The time is "
      parameter( prompt = "The time is ")                          <<<<<<
      CHARACTER datebuf*27
      CALL DATELINE(datebuf)
C>>>>>DISPLAY prompt,datebuf[19:9]
```

```
      print *, prompt,datebuf(19:19+9-1)                              <<<<<<
      STOP
      END
```

When TIME77 is compiled by the HP FORTRAN 77/3000 compiler, it produces the following listing:

```
:FTN TIME77

 PAGE    1  HEWLETT-PACKARD   HP32116X.00.02
   HP FORTRAN 77  (C) HEWLETT-PACKARD CO. 1984   WED, JAN  9, 1985,  9:25 AM


    1     1.000            PROGRAM time
    2     2.000            SYSTEM INTRINSIC DATELINE
    2     3.000      C>>>>>PARAMETER prompt = "The time is "
                           ^
**** WARNING #  1   WARNING: THIS FEATURE IS HP3000 FORTRAN 77 ONLY (830)
    3     4.000            parameter( prompt = "The time is ")
                                            ^
**** ERROR #  1   INCONSISTENT PARAMETER TYPE (153)
    4     5.000            CHARACTER datebuf*27
    5     6.000            CALL DATELINE(datebuf)
    5     7.000      C>>>>>DISPLAY prompt,datebuf[19:9]
                           ^
**** WARNING #  2   WARNING: THIS FEATURE IS HP3000 FORTRAN 77 ONLY (830)
    6     8.000            print *, prompt,datebuf(19:19+9-1)
    7     9.000            STOP
    8    10.000            END

   NUMBER OF ERRORS =     1   NUMBER OF WARNINGS =    2
   PROCESSOR TIME   0: 0: 1   ELAPSED TIME     0 : 0: 8
   NUMBER OF LINES =     10


PROGRAM TERMINATED IN AN ERROR STATE.  (CIERR 976)
```

(Characters in columns 72 and greater are dropped by the compiler when it produces its listing, so the left angle brackets generated by the conversion utility do not appear. They do, however, remain part of the new source file.)

The compiler has detected one error. To correct it, the variable prompt must be explicitly typed. Here is TIME77 with the correction.

```
      PROGRAM time
      SYSTEM INTRINSIC DATELINE
C>>>>>PARAMETER prompt = "The time is "
      CHARACTER prompt*12                      ! Explicitly type prompt
      parameter( prompt = "The time is ")                             <<<<<<
      CHARACTER datebuf*27
```

```
      CALL DATELINE(datebuf)
C>>>>>DISPLAY prompt,datebuf[19:9]
      print *, prompt,datebuf(19:19+9-1)                              <<<<<<
      STOP
      END
```

Again, TIME77 is compiled with the HP FORTRAN 77/3000 compiler, producing the listing below, which is followed by a run of the program. Notice that two warnings are generated by the compiler. These do not affect the running of the program, but merely flag those constructs specific to HP FORTRAN 77 for the 3000. You can turn off generation of these warnings with the compiler directive $STANDARD_LEVEL SYSTEM.

```
:FTNGO TIME77

 PAGE    1  HEWLETT-PACKARD   HP32116X.00.02
   HP FORTRAN 77  (C) HEWLETT-PACKARD CO. 1984   WED, JAN  9, 1985,  9:32 AM


    1     1.000            PROGRAM time
    2     2.000            SYSTEM INTRINSIC DATELINE
    2     3.000      C>>>>>PARAMETER prompt = "The time is "
                           ^
**** WARNING #  1   WARNING: THIS FEATURE IS HP3000 FORTRAN 77 ONLY (830)
    3     4.000            CHARACTER prompt*12
    4     5.000            parameter( prompt = "The time is ")
    5     6.000            CHARACTER datebuf*27
    6     7.000            CALL DATELINE(datebuf)
    6     8.000      C>>>>>DISPLAY prompt,datebuf[19:9]
                             ^
**** WARNING #  2   WARNING: THIS FEATURE IS HP3000 FORTRAN 77 ONLY (830)
    7     9.000            print *, prompt,datebuf(19:19+9-1)
    8    10.000            STOP
    9    11.000            END


   NUMBER OF ERRORS =      0   NUMBER OF WARNINGS =     2
   PROCESSOR TIME   0: 0: 1   ELAPSED TIME       0: 0:10
   NUMBER OF LINES =     11


END OF PROGRAM

END OF PREPARE


The time is   9:32 AM

END OF PROGRAM
```

CUSTOMIZING COMMAND FILES

SECTION 5

Since the actual conversion instructions reside in ASCII files, they can be customized to your specifications. You can add new commands to the existing files, or create new command files that fit your own particular needs. The three supplied command files are FTNCMDS, FTNFREE, and FTNOPTN. If you modify any of command files, we recommend making a copy of the file and making the modifications in the copy. You can then input the new file as the third parameter to the UDC CONVERT.

This section describes how the commands of the command files work. You do not need to read this section unless you plan on creating new command files or adding more transformations to the existing package.

The command files use these commands:

```
R -- perform the replacement, at most once per line

N -- perform the replacement, but don't do any more replacements on this line

G -- perform the replacement everywhere it occurs on the line

D -- if the line is found, delete it

S -- if the line is found, perform the replacement and set an internal flag

T -- if the expression is found and the flag is already set then perform
     the replacement and clear the flag

* -- this character at the start of a line indicates a comment
```

The syntax of the R, N, G, S, and T commands is:

command/search expression//replace expression/

EXAMPLE: R/boy//girl/

The first occurrence of *boy* (upper or lower case) in each line is replaced with *girl*.

The slashes in the example are the search string delimiters. There are two slashes between the search and replacement strings. If one is omitted, the search is performed but no replacement done.

The syntax for the D command is:

D/*delete expression/*

EXAMPLE: D/boy/

If the string *boy* (upper or lower case) is found, delete that line.

The set of commands used in search expressions differ from those used in replacement expressions. In particular, some of the same command characters have different meanings depending on whether they are part of the search expression field or the replace expression field.

Search string commands:

Search string commands are those that appear on the left side of the command.

The following characters are interpreted as commands if they precede a search string, unless themselves preceded by a backslash character (\).

Position Expressions

^	search string starts at beginning of line
$	search string ends at end of line
^*nnn*	search string starts at column *nnn*
$*nnn*	search string ends at *(last column - nnn)*
<*nnn*	search string starts in any column before *nnn*
>*nnn*	search string starts in any column after *nnn*

\ Don't interpret the next character as a command

Tag Fields

{expn} "Remember" the text matched by the expression (*expn*) for use in the replacement string.

a&b Match a string delimited by evenly nested occurrences of *a* and *b* (which can be any two characters), and also "remember" the contents for use in the replacement string.

EXAMPLE:

```
R/(&)//[wow]/
```

This command replaces any set of characters delimited by parentheses with the string *[wow]*. For example, if a source line contained the string *(hi(people))*, the conversion utility would replace it with *[wow]*.

Character Classes

[set] Match any character in *set*, which is defined as:

- a specific list of characters (as, for example, `[abde]` or `[ab012&"@]`); or
- a range of characters, the first separated from the last by a hyphen (as, for example, `[a-z]` or `[0-9]`); or
- a mixture of the two.

`[^set]` Match any characters ***except*** those in ***set***, which is as defined above.

EXAMPLES:

`[a-gxyz]` is the set *abcdefgxyz*
`[a-zA-Z]` is the complete set of alphabetic characters
`[^0-9]` is anything but a digit

Closures

*	Match zero or more occurrences of the preceding character or character class (but specifically do not match anything else).
*nnn	Match exactly *nnn* occurrences of preceding character or class.
+nnn	Match at least *nnn* occurrences of preceding character or class.
?nnn	Match up to *nnn* occurrences of preceding character or class.

EXAMPLE:

The conversion utility puts right angle brackets (>) in columns 2 through 6 of the lines which it converts. If we no longer wanted the lines which the conversion utilities convert we could use the following command to delete these lines. This command deletes each line in which it finds exactly five occurences of the right angle bracket (>).

```
D/>*5/
```

Replacement String Commands

Replacement string commands are those that appear on the right side of the R, N, G, and S commands.

Tag fields

&*n*	Drop tag field *n* into the replacement string at current position. If *n* is not specified, the entire search string is used.
>*n*	Substitute the *n*th tag field matched and shift it to upper case.
<*n*	Substitute the *n*th tag field matched and shift it to lower case.

Fill commands

*x	Change the fill character to *x* (the default is the space character).
^*n*	Fill with fill character up to column *n*.
$*n*	Move the rest of the replacement string so it terminates at column *n*.

EXAMPLES:

The following command uses a replacement string command to move everything found starting at column 2 of the current line to column 7 if an alphabetic character is found in column 2. Columns 2 through 6 are then filled with spaces.

```
R/^2[a-zA-Z]//^7&/
```

`R/^boy//girl/`	replaces *boy* only if it occurs at the beginning of a line.
`R/b.y//girl/`	replaces *boy*, *bay*, *buy*, *bny*, etc.
`R/^6BEGIN//^9BEGIN/`	moves *BEGIN*, if found starting in column 6, to column 9, but does not affect *BEGIN* found anywhere else in the line.

Here is an example conversion file, with comments explaining what it does.

```
* This file converts free format to fixed format FORTRAN.
*
* If the string "$CONTROL FREE" is found, delete that line:
D/$CONTROL FREE/
* If a line begins with a series of alphanumeric characters,
* followed by a space, delete them (or, if the line begins with just
* a space, delete the space).  This removes blank spaces in column 1
* or the FORTRAN/3000 "sequence fields," which have no counterpart
* in HP FORTRAN 77/3000.
R/^[a-zA-Z0-9]* ///
* If a line begins with a #, replace it with a C.
N/^\#//C/
* If a line begins with one or more digits followed by a space,
* then move everything that follows the space to the right of
* column 7 (this moves all statements preceded by statement
* numbers from whatever column they are in to column 7).
R/^[0-9]+ //&^7/
* If a line begins with anything other than a digit or a dollar
* sign, move that to the right of column 7.  This moves all
* statements, except compiler options and statements with
```

```
* statement numbers, to column 7.
R/^[^0-9$]//^7&/
* The T and S commands are used in this order to "remember"
* an & was found on the previous line and to put it on the next line.
* This causes the FORTRAN/3000 construct of an ampersand at the end
* of a line indicating a continuation line conform to the HP FORTRAN 77/3000
* construct of a character in column 6 indicating a continuation line.
T/^6 //\&/
S/\&$///
```

ADDING TO COMMAND FILES

SECTION 6

You can create a new command file that in one pass both converts free-format source to fixed format and performs all the transformations.

Here is how to do it:

1. Build a new command file (called, in this example, COMM):

```
:BUILD COMM;REC=-88,1,F,ASCII
```

2. Copy the command file FTNFREE.PUB.SYS into the new command file:

```
:FCOPY FROM=FTNFREE.PUB.SYS;TO=COMM
```

3. Join the command file FTNCMDS.PUB.SYS to the end of the new command file:

```
:RUN TDP.PUB.SYS
/T COMM
/J FTNCMDS.PUB.SYS 1/LAST TO LAST BY 1
/K
/Purge old COMM? Y
/E
```

Now a free-formatted FORTRAN/3000 source file can be processed with this new command file, causing in one pass both the conversion to fixed format and the performing of all the transformations. Use this command to perform both tasks:

:CONVERT,*oldsource*,*newsource*,COMM

Here is another example of how to create a customized command file.

The command file FTNCMDS (and any command file created from it) puts left angle brackets (<) in columns 72 to 77 of lines it converts. If you wish to remove these, you can create a command file that does the task. This new command file should consist of the following line ***and nothing else***:

```
R/^72\</// /
```