LITTLE Newsletter Number 5
User Information Concerning
LITTLE-to-FORTRAN Translator.

J. T. Schwartz
November 29, 1971

1. The LITTLE-to-FORTRAN translator will be maintained as an UPDATE 'oldpl', catalogued as LTF1077. A binary version of this file may also be catalogued subsequently as LTFE1077. Proper use of the translator should yield a FORTRAN text with few FORTRAN errors.

2. The program is an adaptation of the (new) LITTLE scanner-macroprocessor. It accomplishes translation in four successive phases, with each of which a group of macros may be associated. Normally only the input to the first phase and the output from the last phase are printed. If one wishes to examine the translation process in more detail, the input to the intermediate phases may be displayed by turning sense switch 1 on.

3. A group of macros must be supplied to each phase of the translation process; though of course any such group may be null. The macro-tables are re-initialized at the beginning of each translation phase. The input conventions are as follows: a deck to be translated must initially be present on TAPE3; a macro-file on TAPE4. The macro file consists of four separate macro groups, separated by cards containing the character '*' in column 1. Thus the minimum possible macro file consists of four such cards. The final output appears on TAPE3, which is rewound, and ready for punching, or for printing using COPYSBF(TAPE3,OUTPUT).

4. The overall sequence of phases is as follows.

Phase 1: detects statements of the form

NAME1 NAME2 <text1> = <text2>.,

and converts them to

ZQNAME1 (NAME2<text1>,<text2>).,

an appropriate macro ZQNAME1 will then give the field insertion operation that is desired. LITTLE reserved words such as CALL, DATA, etc., are exempt from this transformation. 'SIZE' cards are transformed into LITTLE-format comments.

Occurrences of 'NAME1 NAME2' in other positions within a statement should be expanded into whatever field extract operation is desired by supplying appropriate macros. LITTLE keywords should be changed to corresponding FORTRAN keywords by supplying appropriate macros.

Phase 2: Performs macro expansion in the normal manner.

Phase 3: Detects labels and GO-TO statements, and makes certain related transformations of text. LITTLE 'named' labels are transformed into FORTRAN 'numbered' labels, and comments recording the correspondence between named and numbered labels inserted into the text. Note in this connection that 'numbered labels' of the form /number/ will be processed in the same way, except that no comment will be inserted. This is convenient for labeling FORMAT statements inserted into the ultimate FORTRAN text. Cf. the remark concerning generated symbols which follows below.

Phase 4: Revises the phase 3 output to force conformity with FORTRAN column conventions. Note that macroprocessing may also be done in this phase. LITTLE terminators '.,' are stripped off.

5. Deduced macroformats. If a name is declared in phase 1 as a macro with arguments, but called without any arguments being supplied, a left parenthesis will be inserted following the occurrence of the name, and a right parenthesis at the next following occurrence of the terminator '.,'. This can be used to transform LITTLE keyword-opened statements to corresponding FORTRAN forms. For example, if the phase 1 macro

+*WRITE(A,B) = FWRITE(A,B)**

and the phase 2 macro

```
+ FWRITE(A,B) = WRITE(1,1000) B
```

are supplied, then any LITTLE 'WRITE' statement of the form

```
WRITE 1,ARRAY.,
```

will be transformed during phase 1 into

```
FWRITE(1,ARRAY).,
```

which during phase 2 can be changed to

```
WRITE(1,1000) ARRAY.,
```

This last will appear in the ultimate FORTRAN as

```
WRITE(1,1000) ARRAY
```

6. Additional remarks. Since four separate phases of macroprocessing are possible, appropriately planned macros can accomplish quite useful transformations.

Numerical FORTRAN labels generated from named LITTLE labels will start with 1 in each SUBR, and proceed progressively. Other numerical labels may be generated by macros containing the generating symbols ZZYA-ZZYZ. (See an earlier newsletter concerning the use of these symbols in macros.) In order to avoid label overlap, the counters for these symbols are started 1000 units apart. That is ZZYA-ZZYE will generate 1, 2, 3, ZZYF-ZZYG will generate 1001, 1002, 1003, etc.