

INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Scientific Programming Dept.
ICT 1900 Series

Fortran Note 12
20.9.65.

General Analysis Routines in the FORTRAN IV Compiler

This note describes the dictionary search routine, the 'field extract' routine and the List System data used in the FORTRAN IV Compiler

RECOGNITION OF STATEMENT TYPES

When it is required to discriminate among the various statement types, the complete statement is in some area of store (either STATEMENT or CIMAGE) in character form, with a pointer (POSI) pointing to the first character of the statement name.

For all statements except Assignment statements, Statement Function definitions and some Logical IF statements a routine STS is used to differentiate among the different statement types.

With this routine the first few non-space characters of the statement name are compared against a "Dictionary" to determine the type of statement.

On entry to STS, X2 points to the first character of the Statement name.. On exit, X2 points to the character following the name found and X3 contains a small integer which indicates what dictionary entry was found. If no dictionary entry is found, on exit from STS X2 will be unchanged and X3 will be zero.

e.g. For the statement

DOUBLE PRECISION X

on entry to STS, X2 should be pointing to 'D'. On exit from STS, X2 would be pointing to the character (space or non-space) following 'N'. X3 would be a small integer indicating which dictionary entry was found. Although in theory it is necessary to compare the whole statement name against a dictionary item, in practice only the first few characters of the longer statement names are compared.

For example, the statement

INTEGER X

is recognised as "INTEGER" by searching for a dictionary item INT rather than for the complete name.

Associated with every dictionary item is a number which tells how many non-space characters need to be ignored following recognition to bring the character pointer to the end of the name.

The dictionary therefore consists of two tables - a 'Compare' table, and an associated 'Ignore' table.

These have the following form:

1. Compare Table

1st word - An Integer which is one less than the number of items in the "Dictionary List". This consists of a series of dictionary items each separated from the next by a space character.

The last item in the list is terminated by a space character. These dictionary items may be any length up to the length for total comparison for the particular statement name.

2. Ignore Table

This is a string of characters (one character per dictionary item) which gives the number of non-space characters of the original statement that are to be ignored following recognition of the dictionary item. The value of the character will be zero if a complete name comparison is made for a particular dictionary item. For example, in the case of INTEGER X the Dictionary item is INT, and the associated Ignore Table character is "4".

The Ignore Table string of characters is stored in the reverse order to the order of items in the Dictionary List, i.e. the character associated with the last item in the Dictionary List is the first item in the Ignore Table and vice versa.

In the FORTRAN Compiler, there are five different dictionaries which are used at different times (i.e. within the Program Description, between program segments, at the beginning of a Program segment, within a MASTER, FUNCTION or SUBROUTINE segment, or within a BLOCK DATA segment).

Before the routine STS is entered to make the dictionary search, the address of the start of the appropriate 'Compare Table' and the address of the start of the associated 'Ignore Table' must be placed in registers LSTP and ISTP of common area LC3.

The value of X3 which is returned upon exit from STS is the number of the entry in the Ignore Table corresponding to the Dictionary Item found. Thus if the item was the last item in the Dictionary List (and therefore the first character in the Ignore Table), the value of X3 would be '1' on exit from STS. If no item is found (i.e. the statement type is not in the current dictionary) the value of X3 on exit from STS would be zero.

X3 on exit from STS can be used as a modifier to branch to the section of the compiler dealing with the type of statement found.

Between the exit from STS and the execution of the appropriate branch instruction, the pointer PCSI is reset to the current value of X2. Thus at the beginning of the analysis of any particular statement (excluding Assignment, Statement Function and some Logical IF statements) PCSI is known to point to the first character following the name of the statement.

Assignment statements, Statement Function definitions and Logical IF statements of the form "IF(1) Assignment" are not detected by the STS mechanism.

These statements are distinguished from all the other statements by the fact that they all have an "Equals" character at level zero and do not have a "comma" at level zero anywhere following the equals. Level indicates the number of ("characters which have been found without a corresponding") character. Thus in the following character string "equals" is at level two.

$$A + (B + (C = D))$$

whereas in the following example, "Equals" is at level zero

$$A = (B + (C + D))$$

The DO statement is the only statement which has "=" at level zero and also has "," at level zero following the equals.

The "status" of a statement (number of "equals" at level zero and "commas" at level zero following the "equals") is determined at the time the statement is formed in STATEMENT and is available for any statement in a register STAT (of Common area LC3).

For Assignment Statements, Statement Function definitions and Logical IF statements of the form "IF(1) Assignment", the value of STAT is one. For all other statements (except DO) the value of STAT is zero. For DO, the value of STAT is two or three, but this fact is not used in the compiler at present.

Statement Analysis (GETFIELD)

For all statements except FORMAT, the basic routine used to ascertain the structure of the statement is GETFIELD.

On entry to this routine (at GFE1), PCSI is pointing to the first character of a "field" which is to be found. On exit, PCSI is pointing to the first character following the field terminator.

At this point, the field is in a store area starting at register NAME. The length of the field is held as a counter in the register LENG. The field terminator is in the register TERM (and in X5). The type of field found is given by the value of X4 as follows.

X4 < 0 No field.

In this case, POSI was not originally pointing to a letter or the start of a number or a logical constant. On exit, TERM holds the character originally pointed at unless that character started a character string with the form of a logical or relational operator (e.g. .AND.) in which case TERM holds the internal single character representation for that operator. POSI points to the first non-space character following the original character (or logical or relational operator).

X4 = 0 Alphanumeric Field

In this case POSI was originally pointing at a letter. On exit, the store area starting at NAME will contain the character string starting at the letter and terminating with the first non-space, non-alphanumeric character (which will appear in TERM) after that letter. Space characters are ignored in the character string. Thus the field 'AL PH A' is the same as the field 'ALPHA'.

X4 = 1 Floating Point Number

In this case POSI was pointing at the first character of a character string representing an unsigned floating point number. On exit NAME --> NAME + 1 (and X6, X7) contains the normalised internal representation of that number. LENG is 2 (in the counter position).

X4 = 2 Small Integer

In this case POSI was pointing to a digit starting a digit string representing a small integer (< 4096). The terminator of this string is not a point (.) unless this starts a logical or relational operator in which case the terminator is the complete operator (in internal single character form).

On exit NAME (and X7) contains the internal binary representation of the integer and LENG is 1 (in the counter position).

X4 = 3 Large Integer.

This is the same as for X4 = 2 except that the value of the integer is in the range 4096 to 8388607.

X4 = 4 Double Precision Number

In this case POSI was pointing to the first character of a character string representing an unsigned double precision number.

