INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Programming Languages Division — FORTRAN NOTE19

I.C.T. 1900 Series — Date 13.1.66.

Program Description and Intersegment Statements.

This FORTRAN NOTE describes the program description and intersegment statements which are shortly to be implemented in A.S.A. Fortran. It supersedes the appropriate sections of FORTRAN NOTES 2,3 and 7.

GENERAL

This document describes the Program Description and intersegment statements which will appear in A.S.A. FORTRAN. It is intended to retain the existing Program Description statements for a short time as well. Thus existing programs may still be compiled but should be modified as soon as possible to conform to the new program description formats. All the statements are written in the normal FORTRAN manner in columns 7 to 72 of a line, spaces being ignored (except in file and subfile names). However, no continuation lines are allowed for these statements.

A complete program is made up of a program description, intersegment statements and source or semi-compiled program arranged in the following manner.

a) Intersegment statements (see note below).
b) Program Description.
c) Source or semi-compiled program mixed as required with intersegment statements.
d) The statement FINISH.

NOTE: The first statement presented to the compiler should define the listing peripheral and the listing mode required, e.g. LIST (LP). By default SHORTLIST (LP) is assumed. The second statement should define the output medium, but is, at present, redundant since the type of compiler used automatically defines the output medium.

PROGRAM DESCRIPTION.

The Program Description may be considered as a segment introduced by one of the statements PROGRAM or SEGMENTS and terminated by the statement END. It must be the first segment read in any A.S.A. FORTRAN compilation but may optionally be preceded by intersegment statements. The program description is compiled as a "Master" segment with a "Force-in" Bit.

1. The PROGRAM Statement.

The PROGRAM statement introduces the Program Description and specifies the name of the object program. It has the forms:-

PROGRAM (n)

PROGRAM (nc)

where "n" is the name to be given to the object program and "c" is an accounting code.

"n" consists of 4 alphanumeric characters starting with a letter. "c" consists of up to 8 alphanumeric characters and is optional, but may be insisted upon at some installations which may use it to charge the user for the compilation and running of a program.

The PROGRAM statement is essential if a complete program is to be consolidated into a loadable, semi-compiled object program.

2. The SEGMENTS Statement.

The SEGMENTS statement is used instead of the PROGRAM statement when one or more segments are to be compiled, but not consolidated into a loadable program at this stage; i.e. request slip, loader and consolidated leader are not to be output. The statement has the forms:-

SEGMENTS

SEGMENTS (n)

SEGMENTS (nc)

where "n" and "c" are the same as in the PROGRAM statement. The second form is necessary if output is to magnetic tape and the third form is necessary if an accounting code is obligatory.

3. The TRACE and NOTRACE Statements.

The TRACE and NOTRACE statements indicate to the compiler whether to compile in trace mode or not. They take the forms:-

TRACE

NOTRACE

These statements may also appear between segments to compile different segments in different modes. The effect of TRACE and NOTRACE is to set and reset a switch, initially NOTRACE is assumed by default.

4. The OVERLAY Statement.

The OVERLAY statement specifies which segments of the program are not to be in permanent program and which overlay area and unit they are to appear in. It has the form:-

OVERLAY (a,u) s_1, ---, s_n

where "a" is an integer between 1 and 255 inclusive and is the overlay area, "u" is an integer between 1 and 1023 and is the overlay unit, and "s_i" are the names of the segments to appear in this overlay unit of this overlay area. Segments which may appear in an overlay unit are SUBROUTINE segments with not more than 20 dummy arguments.

5. The PRIORITY Statement

The PRIORITY statement specifies the priority to be assigned to the object program and has the form:-

PRIORITY n

where "n" is an integer in the range 1 - 99.

It is only relevant if the Program Description is introduced by a PROGRAM statement.

Omission of PRIORITY implies

PRIORITY 50

6. The Peripheral Statements.

These are the INPUT, OUTPUT, USE and CREATE statements. They describe what peripheral devices are to be used by the object program and they have the general format:-

TYPE n_1, ----, n_k = peripheral information

where:-

"TYPE" is one of INPUT, OUTPUT, USE or CREATE.

INPUT and OUTPUT may be used with any peripheral indicating that the peripheral is to be used only for input or output respectively. USE and CREATE may only be used with magnetic tape, USE incidating that the tape may be used for both input and output, and CREATE indicating that a Scratch Tape is to be opened, re-named and may then be used for both output and input.

"n_i" are the values by which the peripheral device is known within the object program. They are either small integers, 1 to 4095 inclusive, or (MONITOR) which indicates that the peripheral concerned is to be used as the input or output device for TRACE information.

"peripheral information" designates the peripheral device, "p", which is known in the object program by the values "n_i" . It may also give auxiliary information, "a", such as file name, and a qualifier, "q", may also appear to indicate that a special peripheral routine is to be used.

The format is one of

p/q (a)

p/q

p (a)

p

where:-

"p" is the usual code for peripheral devices, e.g. CR0, LP1, MT7.

"q" may only be, at present, "FORMATTED", which may be used in conjunction with magnetic tape to indicate that formatted information is to be recorded on and/or read from it. Omission at "FORMATTED" implies unformatted information.

"a" is used at present only to give the file name of the magnetic tape and consists of up to 12 characters chosen from A to Z, 0 to 9, space and hyphen. In the case of magnetic tape omission of "a" will imply a scratch tape. The auxiliary information will shortly be extended to specify subfile name, retention period and generation number.

7. The COMPRESS Statements.

These statements have the forms:-

COMPRESS INTEGER AND LOGICAL

COMPRESS DOUBLE PRECISION

The first one causes all Integer and Logical variables to be assigned one word of core store instead of two. It should be emphasised that this does not reduce the range of integer constants since although they normally occupy two words, to conform with the ASA specification, only the first of these is used to hold the constant, the second being ignored. Thus in either case integer constants are held in one word length only. The second statement causes all Double Precision variables to be treated as Single Precision (Real) variables and assigned two words of core store instead of four. These two facilities should be used with caution particularly if they affect variables used in COMMON or EQUIVALENCE.

8. The ON and OFF Statements.

These statements may be used to switch bits of word 30 of the compiler on or off and have the form:-

ON n_1, - - - - , n_n

OFF n_1, - - - - , n_n

where "n_i" are integers between 0 and 23 and indicate the bits of word 30 to be switched on or off.

These statements may also appear between segments. They are intended for use only by the compiler writers.

9. The LEADER Statements.

This statement, which has the form:-

LEADER

causes the Consolidated Leader to be listed as described in FORTRAN NOTE 16.

10. The COPY and NOCOPY Statements.

These statements, which have the forms:-

COPY

NOCOPY

cause any acceptable semi-compiled program to be copied or not copied, depending on the statement. These statements will be ignored when output is to magnetic tape as everything must be copied.

They may appear between segments to copy segments as required.

Omission of either normally implies COPY.

11. The END Statement.

This statement, which has the form:-

END

must terminate the Program Description.

INTERSEGMENT STATEMENTS.

The following are the only statements which may appear between segments.

1. The LIST and SHORTLIST Statements.

These statements indicate the type of listing required and on what device it should be output. They have the forms:-

LIST

LIST (p)

SHORTLIST

SHORTLIST (p)

where "p" is LP or TP for listing on a line printer or paper tape punch respectively. Omission of "p" implies LP.

LIST gives a full listing of source (except semi-compiled) plus diagnostic information and a listing of control statements. SHORT LIST gives a listing of control statements, segments headings and diagnostic information.

The form:-

LIST (TP)

will be interpreted as SHORT LIST(TP).

By default SHORTLIST (LP) is assumed.

LIST or SHORT LIST should be the first statement of any program, to indicate which peripheral to list on. They may also appear between segments so that some segments may be full listed and others short listed.

2. The TRACE and NOTRACE Statements.

See Program Description number 3.

3. The COPY AND NOCOPY Statements.

See Program Description number 10.

4. The ON and OFF Statements.

See Program Description number 8.

5. The READ FROM Statement.

This statement specifies the peripheral from which the subsequent source segments are to be read and has the forms:-

READ FROM (p)

READ FROM (MT, f)

where "p" is CR or TR for reading from a card reader or paper tape reader respectively and "f" is information about the magnetic tape file or subfile to be read, in a form yet to be specified.

6. The SEND TO Statement.

This statement specifies the output peripheral to receive the object program. Versions of the compiler that permit output to only one fixed slow peripheral (paper tape or cards) will ignore it. It has the forms:-

SEND TO (p)

SEND TO (MT, f)

where "p" is one of TP, CP or MT indicating output to a paper tape punch, card punch, or magnetic tape respectively; and "f" is information about the output file in a form yet to be specified.

This statement must immediately precede the PROGRAM or SEGMENTS statement of the Program Description.

7. The FINISH Statement.

This statement has the form:-

FINISH

and has the following effects.

1) If there have been errors the compiler is suspended and the console message:-

HALTED ZZ

is typed. It is not then possible to continue compilation.

2) The mode of compilation is switched to "selective"; i.e. segments read after a FINISH will only be accepted if they have either been called for or have a "force-in" bit.

3) If there are any unsatisfied cues the FORTRAN library is scanned (or the leaders read from a LIBRY segment in paper tape output versions).

4) When all cues become satisfied consolidation will be completed and Request Slip, GPL and Consolidated Leader output.

5) If there are any unsatisfied cues remaining after 3) above, the compiler is suspended and the console message:-

NEEDS c

is typed, where "c " is one of the unsatisfied cues. A list of all the unsatisfied cues is output on the listing peripheral.

Compilation may be continued by typing the console message:-

GO # name

where "name" is the name of the compiler.

8. The LIBRARY Statement.

This statement, which has the form:-

LIBRARY

has the effects 1), 2) and 4) given in the FINISH statement (see above). It is useful if a programmer wants his own library read before the standard FORTRAN library.

N.F. Bearne.

APPENDIX I

This appendix summarises the statements available. Any parameter within square brackets is optional.

Program Description Statements.

1) PROGRAM (n[c])
2) SEGMENTS [(n[c])]
3) TRACE
   NOTRACE
4) OVERLAY (a,u) s_1,---, s_n
5) PRIORITY n
6) INPUT n_1, ----, n_n = pn [/q] [(a)]
   OUTPUT n_1, ---, n_n = pn [/q] [(a)]
   USE n_1, ---, n_n = MTn [/q] [(a)]
   CREATE n_1,---, n_n = MTn [/q] [(a)]
7) COMPRESS INTEGER AND LOGICAL
   COMPRESS DOUBLE PRECISION
8) ON n_1, --- , n_n
   OFFn_1, --- , n_n
9) LEADER
10) COPY
    NOCOPY
11) END

Intersegment Statements.

1) LIST [(p)]
   SHORT LIST [(p)]
2) TRACE
   NOTRACE
3) COPY
   NOCOPY
4) ON n_1, ---, n_n
   OFF n_1, ---, n_n
5) READ FROM (p[,f])
6) SEND TO (p[,f])
7) FINISH
8) LIBRARY