INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Programming Languages Division
I.C.T. 1900 Series

FORTRAN NOTE 20
18.1.66

Provisional specification of the 1900 Fortran compiler

This specification relates to the pre-release versions of the ASA Fortran compilers; //ZFAM (magnetic tape version) and //ZFAP (paper tape version). The compilers due for release on 1.3.66 will be named //XFAM and //XFAP and will not necessarily be to the same specification as given in this note.

This series of notes is intended for reference by the writers of the compilers and associated software. This particular note and Fortran Note 19 is being given a wider circulation than usual because they give details of interest to programmers using the ASA Fortran compilers.

- DRAFT -

(PROVISIONAL)

#ZFAM

#ZFAP

TITLE

1900 Fortran compiler

ISSUE

1

HARDWARE REQUIREMENT

	#ZFAM	#ZFAP
Core store:	12000 words	13000 words
Input peripheral:	1 paper tape reader and/or	1 paper tape reader and/or
	1 card reader and optionally	1 card reader and optionally
	1 or more magnetic tape decks	1 or more magnetic decks
Output peripheral:	1 magnetic tape deck	1 paper tape punch
Listing peripheral:	1 lineprinter and/or	1 lineprinter and/or
	1 tape punch	1 tape punch
Other peripherals:	2 magnetic tape decks (Work tape plus program library tape)	None

USE OF PERIPHERALS

Source program and/or semi-compiled program may be input from either paper tape, cards, or magnetic tape. Initially, according to choice, a paper tape reader or card reader is allotted; thereafter the input device is selected dynamically under control of the program. A paper tape or card reader is retained throughout a compilation, even if the bulk of the source is on magnetic tape.

Normally a lineprinter is allotted at the start of the run to receive a listing of the source program and error information, if any. A tape punch may be used as an alternative, in which case a listing will be given in an abbreviated form.

The other peripherals used depend on the version of the compiler.

a) /ZFAN

A scratch magnetic tape is opened at the start of a run and used as a work tape. It is left scratch.

A magnetic tape, normally the program library tape, containing the Fortran library must be available throughout the run.

At the end of a successful compilation a second scratch tape, or a tape with a specified name, is opened, given a specified name and retention period, and used to receive the object program.

b) /ZFAP

A paper tape punch is used to receive the object program. Output will cease on the occurrence of an error, and the punch will be released.

Allocation and release of peripherals:

All peripherals are allocated dynamically as and when required.

A special entry point is provided to release all peripherals. During the course of a run peripherals will only be released when a change is made (e.g. switching from cards to paper tape input).

Since a list is kept within the compiler of all peripherals being used, console allocation and release of peripherals should be avoided.

DESCRIPTION

a) General

The A.S.A. FORTRAN IV compiler will translate FORTRAN segments into standard 1900 semi-compiled form and will optionally attempt to consolidate these segments, and any other semi-compiled segments that may be provided, into a complete object program. The FORTRAN segments should be written in the version of the language described in the "I.C.T. 1900 Manual" (Provisional edition Dec. 1965).

For a complete program the compiler will automatically produce an executable program, complete with General Purpose Loader. For one or more segments not forming a complete program the compiler will produce only semi-compiled segments. At a later stage these segments may be re-input to the compiler along with other FORTRAN and/or semi-compiled segments that together form a complete program.

For the A.S.A. Fortran IV compiler a Program Description must be provided. This is read prior to input of source or semi-compiled program, and enables compile time and object time parameters to be specified by the user. Further details are given under the heading INPUT PARAMETERS.

During compilation, unless specified otherwise, a listing is given on a lineprinter of the source program and error information (if any).

b) Input

Input to the compiler may be on paper tape in the standard 1900 eight-track code and/or on punched cards in the 1900 card code. Source or semi-compiled program may also be input from magnetic tape.

A program basically consists of one or more segments followed by a FINISH statement. There must be one and only one MASTER segment. The program may contain only FORTRAN segments, only semi-compiled segments or a mixture. Segments may be in any order: in particular, the MASTER segment need not appear first.

All of these segments will be incorporated in the object program, except that if two or more segments with the same name are input, only the first one is incorporated. The FINISH statement causes the compiler to arrange for any required segments from the FORTRAN library to be incorporated. This includes segments that are private to the Fortran compiler, e.g. Fortran Input/Output package.

If the user has a series of subroutine and function segments that form a private library they may be read after a LIBRARY statement and should precede the FINISH statement. Before a FINISH or LIBRARY statement the compiler will incorporate into the object program all segments encountered. After a FINISH or LIBRARY statement has been encountered only those segments to which a reference has previously been made are incorporated. Segments in a private library will normally be in semi-compiled form, but they could be in FORTRAN source form.

Any semi-compiled segments input to the FORTRAN compiler may have been previously produced by the FORTRAN compiler or by the PLAN assembler.

The compiler can start reading from either paper tape or punched cards. Wherever there is to be a switch from one medium to the other the programmer should insert one of the following statements:

READ FROM (CR)	Switch from paper tape to cards.
READ FROM (TR)	Switch from cards to paper tape.
READ FROM (MT,.....)	Read subfile from magnetic tape specified and then continue reading from slow peripheral.

As usual, the statements should not start before character position 7 of the line.

c) Output

The Program Description allows the user to specify that either;-

A) the source is to be compiled to produce one or more semi-compiled segments, in which case the output is in semi-compiled form on magnetic tape (//ZFAM) or paper tape (//ZFAP).

or B) the source is to be compiled and consolidated to produce a loadable program as described below.

i) //ZFAM

If no errors are found and the program is complete, the output is a magnetic tape containing an executable program ready to be input by an Executive FIND directive.
If no errors are found but the program is not complete, there is no useful output unless the operator has been requested to force the output of a General Purpose Loader. Then the output will be as above. This action is sensible only if it is known that the missing segments, although referred to, will not be needed when the program is run. If errors are found there is no useful output and consolidation cannot be completed.

ii) //ZFAP

If no errors are found and the program is complete, the output is a length of paper tape containing:

a) semi-compiled segments.
b) runout.
c) request slip and General Purpose Loader.

The request slip and General Purpose Loader are torn off. When the program is to be run they must be input before the semi-compiled segments.

If no errors are found but the program is not complete the output is a length of paper tape containing only semi-compiled segments. If the operator has been requested to force the output of the General Purpose Loader, the output will be as in the last paragraph. This action is sensible only if it is known that the missing segments, although referred to, will not be needed when the program is run.

If errors are found, the output is a length of paper tape containing any semi-compiled segments produced before the first error occurred plus an incomplete semi-compiled version of the segment in which the first error was found.

d) Listing

During the course of compilation a listing will be produced on either a lineprinter or a tape punch. On the lineprinter two forms of listing are allowed; either a short list or a full list, the difference being that, except for segment names, source statements are not printed in the case of short list. Unless otherwise specified short list is assumed and if the tape punch is used then only short list is allowed.

Statements occurring within the program description or between segments are always listed. The occurrence of semi-compiled program will cause one line to be printed giving the segment name e.g.

SEG2B (SEMI-COMPILED)

Additional information is output by the compiler giving the compiler name, date and time run, name and number of instructions of each segment compiled, program name and core store required and the number of errors detected.

e) Error Interpretation

If the compiler detects an error in the input it ceases to output object program but continues to search for errors. Errors discovered by the compiler will be output on the listing peripheral in the form:-

ERROR n IN LAST STATEMENT, LINEp, CHARq

where n is the error number
p is the line number within the statement (first line = 0)
q is the character number (first character = 1) and gives an indication of the last field investigated by the compiler before the error occurred; the actual error may have occurred earlier.

The significance of the error number is as follows:-

ERROR NUMBER	INTERPRETATION
1	Mis-matched parenthesis.
2	Missing parenthesis
3	Some other character found when /) or, was expected.
4	Error in format of arithmetic expression.
5	Constant out of range.
6	Should be an integer variable.
7	Array not declared, or declared twice.
8	First character of name non-alphabetic, or name omitted.
9	Statement too long.
10	Statement incomplete
11	Label set twice or error in columns 1 to 5.
12	Error in label reference
13	Illegal subscript, e.g. too many or too few subscripts.
14	Illegal name, e.g. subroutine has illegal name.
15	Label missing
16	Do label not found.
17	Statement not recognised or out of context.
18	Error in statement other than a source statement.
19	Polish list exhausted.
20	Compiler error.

The above interpretation of errors having a code number in the range 1 to 25 applies regardless of statement type, but the interpretation of error numbers above 25 depend on statement type as follows:-

Statement	Error Number	Interpretation
MASTER	26	Master name in overlay list.
FUNCTION	26	Segment name omitted.
	27	No arguments
SUBROUTINE	26	Segment name omitted.
CALL	26	Not subroutine name.
GO TO	26	Error in statement after GO TO.
	27	Error after) of computed GO TO
DO	26	Do label followed by end of statement.
	27	'=' missing or misplaced.
	28	Parameters not integer expressions?
IF	26	Do Statement in 'Logical IF'.
FORMAT	26	Unlabelled.
READ	26	Unit number not integer constant. or integer variable.
WRITE		
	27	Format reference error
	28	Error in list following.
INTEGER	26	Type already specified.
REAL		
LOGICAL		
DOUBLE P		
COMPLEX		
COMMON	26	Illegal name for common block.
	27	Illegal item in list.
DIMENSION	26	No dimension given.
EQUIVALENCE	26	Dummy argument.
	27	Illegal argument, e.g. constant, statement label, subroutine name.
	28	Argument in list is followed by '(' and it is not an array name.
	29	Argument already equivalenced or otherwise defined so that equivalence is impossible.

DATA	26	Illegal name in variable list, e.g. a subroutine name.
	27	Illegal field in constant list.
	28	Illegal form of complex constant.

When short list mode is specified the first line only of the statement in error is printed prior to the above error message.

If on the occurence of the END statement there exists any labelled statements which have not yet been found then for each one a message will be output, following the END statement, of the form

ERROR n - LABEL l

where n is the error number (see above)

l is the label not found.

INPUT PARAMETERS

The Fortran IV compiler permits a very versatile set of input parameters to be supplied either as part of the program description or as control statements between segments. A full description is given in Fortran Note 19.

A complete job presentable to the compiler is of the form:-

1. Initial Statements. (e.g. LIST(LP))
2. Program Description.
3. Source or semi-compiled segments mixed as required with statements between segments.
4. The FINISH statement.

STORE SPACE USED

#ZFAM	12000 words
#ZFAP	13000 words

PRIORITY

The program as supplied has a priority of 50.

OPERATING INSTRUCTIONS

(a) //ZFAM

1. Load program //ZFAM

2. Activate by one of the messages

GO //ZFAM AT 20 Compile, reading initially from paper tape.
GO //ZFAM AT 21 Compile, reading initially from cards.

A paper tape reader or card reader is allotted and a scratch magnetic tape opened as a work tape.

3. Load the first paper tape or pack of cards on the reader. Ensure that another scratch tape is available to receive the object program and that a tape containing the FORTRAN library is available.

4. Load any further paper tapes or packs of cards or magnetic tapes in the specified order. Whenever there is a change of medium one reader is released and another of the appropriate type is allotted. However, when a magnetic tape is allocated the card or paper tape reader is not released since control will be passed back to the slow peripheral when the magnetic tape has been read.

5. After a successful run, another scratch tape or a named tape is opened and labelled, and the compiled program copied to it. The end-of-run message

//ZFAM HALTED:- COMPILED 'Name' EC

is then output on the console, where 'Name' consists of 12 characters; the first four being the name of the program; the remainder being the charge number or accounting code. The work tape is left scratch.

6. Return to step 2 to compile another program.

(b) //ZFAP

The operating instructions are the same as above with ZFAP in place of ZFAM except that output is to paper tape in the usual manner, (i.e. semi-compiled, followed by the loader if consolidation has been completed successfully).
Magnetic tapes are not used, unless of course they contain input data.

EXCEPTION CONDITIONS

In the following, //ZFA$_j$ stands for either //ZFAM or //ZFAP.

1. //ZFA$_j$ HALTED:- NEEDS 'Segment'

This will occur, if on the occurrence of the FINISH statement, there remains one or more unsatisfied cues, the first of which will appear as 'Segment' above.

Further source or semi-compiled program may then be read after giving the console message:

GO //ZFA$_j$

Alternatively consolidation can be forced by:

GO //ZFA$_j$ 29 (See note (ii))

2. //ZFAj HALTED:- COMPILED 'Name' ZZ

This indicates that errors have occurred. Consolidation cannot be completed but further programs may be compiled by returning to step 2 of the operating instructions. In the case of //ZFAM the output magnetic tape is not allocated and the work tape is left scratch. If //ZFAP was the compiler used then some useful semi-compiled program may have been output.

3. //ZFA$_j$ HALTED:- PD

This indicates that an error has occurred either before or within the program description. It is possible to continue compilation by:

GO //ZFA$_j$

but only for the purpose of checking for errors in the source program.

4. //ZFA$_j$ HALTED:- OV

This will occur when one or more segments declared as overlay segments. It is necessary to use a separate overlay consolidator to produce a loadable program.

5. //ZFA$_j$ HALTED:- TR
//ZFA$_j$ HALTED:- CR
//ZFA$_j$ HALTED:- TP
//ZFAj HALTED:- LP

The occurrence of one of these messages indicates that the peripheral mentioned is not available. When it is made available the run may be continued by giving the message:

GO //ZFA$_j$

The peripheral will then be allocated.

6. //ZFA$_j$ HALTED:- CE

When this occurs it indicates that the compiler has detected a check sum error while reading semi-compiled program.

If this occurs while paper tape is being read move the input tape back to the beginning of the block and type the message:

GO //ZFA$_j$

If it happens again on the same block, mark the tape and terminate the compilation.

With //ZFAM, if there is a checksum error while reading the library from magnetic tape it is necessary to re-start the compilation from the beginning. This may be achieved by giving the message:

GO //ZFAM 27 (See note (iv) below)

and then continuing at step 2 of the operating instructions.

NOTES:

(i) If the compiler expects more input and none has been supplied by the programmer, suspend the compiler and re-activate by the message:

GO //ZFA$_j$ 28

Action is then as for exception condition 1 above.

(ii) If the input is incomplete, consolidation may be forced by the message

GO //ZFA$_j$ 29

Action is then as in step 5 of the operating instructions. If a list of blank cues is required, it should be obtained before the entry at 29.

(iii) If the compiler goes illegal or loops or otherwise comes to an unexplained stop it is recommended that the standard error procedure is carried out(See below).

Restart should then be possible by the message:

GO #ZFA$_j$ 27 (See (iv) below)

and then continuing at step 2 of the operating instructions.

(iv) It is dangerous to 'take' or 'give' peripherals when using #ZFA$_j$ compilers since this is likely to cause the compiler to go illegal. If at the end of a run it is desired to release any peripherals allocated to the compiler it is simply necessary to give the message:

GO #ZFA$_j$ 27

STANDARD ERROR PROCEDURE

If illegal on instruction other than 150 - 157 then output the first 3,500 words of core store on the lineprinter. This information will be of use later to the compiler writer.

If illegal on instruction in the group 150 - 157 then the console log is usually sufficient to explain the cause.

OPERATING INSTRUCTIONS (FORTRAN Object Programs)

1. Load the program. If it is on paper tape, load a paper tape containing the FORTRAN library on the same reader.

2. Activate the program by the message

GO #name AT 20

or as specified by the programmer. 'name' represents the name of the program.

3. If the program was compiled in trace mode, a steering list may be read at any time the program is suspended. Load the list on a card reader or paper tape reader and type the message

The list is read, the program is suspended and the message

#name HALTED:- ES

is output.

4. No trace information is output unless trace is switched on by the message

ON #name 0

Trace may be switched off by

OFF #name 0

5. If the program was compiled in trace mode, further information about the last 100 statements obeyed may be output if the program is suspended and then reactivated by the message

GO //name AT 29

The program will suspend and output the message

//name HALTER:- EP

This action should always be taken at the end of a run for a traced program, unless the program has deleted itself.

EXCEPTION CONDITION (FORTRAN Object Program)

N.B. This section applies only if the program was compiled in trace mode.

1. After any error stop, the program should be re-activated as in step 5 of the operating instructions.

One special type of error stop outputs the message:

//name HALTED:- AE

It should be treated as above.

18th Jan 1966. L.R. FAIRBROTHER.