V. Taylor

INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Programming Languages Division

FORTRAN NOTE 26A

July, 1967.

Since the issue of Fortran Note 26, //XFAE has been re-organised fairly drastically, and some other modifications have been made. Except where otherwise stated, Fortran Note 26 is still applicable to the new version however. This Note describes the principal changes.

1. General

The purpose of re-organising //XFAE was to reduce its overall size from 11K to not more than 10K. This was made necessary by the discovery that on some 16K machines, Executive might require 6K of store.

The 10K version will necessarily be slower than the original, because normally it will be necessary to read an overlay twice per segment of program compiled.

2. Overlays in the 11K version

There were two overlays as follows:

(1/1) FZ2D, FY2D

(1/2) FG2A, FK2A, FM2A, FN2A, FR2A

Permanent program contained the following segments:-

NAMEXFAE, FA2D, FB2D, FC2D, FE2A, FF2A, FH2A, FI2A, FJ2A, FL2D, FP2A, FQ2A, FS2A, FT2A, FU2A, INTFC.

Where the segment name ends in A, this implies that the segment is identical to the corresponding segment in the magnetic tape compiler.

Overlay (1/1) : FZ2D dealt with the Program Description.

FY2D dealt with basic initialisation, allocation and release of peripherals, READ FROM and SEND TO statements, and the FINISH statement.

Overlay (1/2) : FG2A dealt with EQUIVALENCE and DATA

FK2A dealt with 'Type' statements, and with DIMENSION and COMMON statements.

FM2A, FN2A and FR2A formed the heart of the compiler, dealing with such things as the Polish list and the generation of instructions.

3. Operation of the 11K version

Overlay (1/1) was read in initially and would normally remain in store throughout the processing of the SEND TO (if any), the Program Description, and any initial READ FROM's. At the beginning of each Fortran segment, overlay (1/2) would be read in unless already in store. In practice it would normally remain in store until the FINISH statement was found, unless peripheral switching (which requires overlay (1/1)) occurred meanwhile.

If a SEND TO statement or a statement introducing the Program Description occurred out of context, overlay (1/1) would be read in to deal with it. In fact a SEND TO out of context would be ignored; anything else out of context would result in an error flag.

4. Overlays in the 10K version

This version contains four overlays as follows:

(1/1) FW2D

(1/2) FZ2D

(1/3) FG2A, FK2A, FX2D, FY2D

(1/4) FH2D, FI2A, FJ2D, FM2D, FN2D

Permanent program contains the following segments:

NAMEXFAE, FA2D, FB2D, FC2D, FE2D, FF2D, FL2D, FP2A, FQ2A, FR2A, FS2A, FT2A, FU2A, FV2D, INTFC.

The practice has been continued of changing the last letter of a segment name to D when the segment is no longer identical to the corresponding segment in the magnetic tape compiler. However segments with names ending in D are not necessarily the same in the 10K and 11K versions of the EDS compiler.

Overlay (1/1) : FW2D is a new segment containing the compiler initialisation routine, and the processing of the SEND TO statement. As this is the smallest overlay, there is plenty of room for the SEND TO processing to be expanded, which will almost certainly be necessary in due course.

Overlay (1/2) : FZ2D deals with the Program Description. As it contains a large number of Lower Presets, it was found necessary to put it in a separate overlay to save space.

Overlay (1/3) : FG2A and FK2A are, as their names imply, the same as the corresponding segments in the 11K version. Between them they deal with EQUIVALENCE, DATA, DIMENSION, COMMON and all 'Type' statements.

FX2D is a new segment containing some of the material originally to be found in segment FF2A of the 11K version. It deals with MASTER, BLOCK DATA, FUNCTION and SUBROUTINE statements.

FY2D deals with READ FROM and FINISH statements.

Overlay (1/4) : FH2D, FI2A and FJ2D contain processing that was done in permanent program in the 11K version by the corresponding segments FH2A, FI2A, and FJ2A. FH2D deals in addition with PAUSE, STOP and RETURN statements, which were originally part of segment FF2A.

FM2D and FN2D are basically the same as the corresponding segments in the 11K version. However, like FH2D and FJ2D, they have lost a number of routines which had to be put in permanent because they were required by more than one overlay.

Permanent program : This includes segment FR2A which was originally overlaid, but is now used by more than one of the overlays. Segment FV2D is a new segment containing miscellaneous routines which are used by more than one overlay in the 10K version.
Segments FH2D, FI2A, and FJ2D are no longer in permanent program, and segment FF2D is somewhat smaller, containing only the EXTERNAL and FORMAT statements and the generation of Function and Subroutine prologues.

5. Communication between overlays

Reading of overlays is accomplished by the standard overlay package %EROL which is incorporated in the compiler.

For each of the four overlays (1/1), (1/2), (1/3), (1/4) the compiler contains a cued routine. These are GET1, GET2, GET3 and GET4 respectively. Each contains a BRING instruction which causes %EROL to ascertain whether the relevant overlay is already in store or not, and to read it in if it is not. An overlay which is already in store is never read in on top of itself.

GET1, GET2, GET3 and GET4 are used in the following circumstances:

(1) GET1 is used immediately after entering the compiler so that the compiler initialisation can be performed. It is also used when a SEND TO is detected by the routine FMAS2; however normally overlay (1/1) will still be in store when the SEND TO is detected.

(2) GET2 is used when one of the statements which can introduce the Program Description is detected by FMAS2.

(3) GET3 is used in the following cases:

a) On detection of a READ FROM by FMAS2.

b) On detection of a FINISH by FMAS2.

c) On detection by FMAS2 of a statement which can introduce a Fortran segment.

d) On detection by the routine STATE in FE2D of any of the Fortran statements dealt with by segments FG2A or FK2A.

e) When the compiler has been entered at word 27.

f) When a data subfile has just been read from EDS and the compiler wishes to return to segment FY2D to see whether the READ FROM has been fully processed.

(4) GET4 is used in the following cases:

a) On detection by the routine STATE of any statement in the 'Compare' table other than an EXTERNAL or FORMAT statement or other than a statement which requires overlay (1/3).

b) On detection by STATE of any arithmetic type statement.

6. Operation of the 10K version

It is clear from the above that the compiler always checks that the correct overlay is in store before beginning to process any statement which may require that overlay. The new system does not therefore impose any new restrictions on the ordering of statements, over and above those imposed by any of the Fortran compilers.

On the other hand, statements appearing out of context might well cause wastage of time, since it might be necessary to read in another overlay in order to process them.

It is particularly important to adhere to the rule laid down in the Fortran manual that **all** 'Type', DIMENSION, COMMON, DATA and EQUIVALENCE statements in a segment should be grouped together at the beginning. EXTERNAL and FORMAT statements, however, may appear anywhere in the segment without loss of efficiency.

If these rules are followed, the compiler will normally read (1/3) at the start of each segment, and (1/4) in the course of processing the segment. Two overlays would thus be read for every segment compiled, once the SEND TO and Program Description have been disposed of.

The time required to read an overlay is uncertain at present, but should be much less than one second.

7. Notes on miscellaneous points

1) It was found necessary to duplicate the routine APER by placing identical versions of it in the two overlays (1/1) and (1/3). This was preferable to putting it in permanent, which would have required another 90 words or so of store. Both routines are called APER within their own segments, but the version in (1/1) has been given the cue APERW since it is called from outside.

2) It was found necessary to alter the order of the entries in the tables TAB22, MST2 and MIT2 in segment FE2D so that statements requiring overlay (1/3) could be distinguished from those requiring (1/4) by testing the modifier.

3) A new marker DOIND has been introduced in segment FE2D, whose function is to avoid the call to DO2E (in segment FI2A) unless the correct overlay (1/4) is in store. Since DO loops should never end on non-executable statements it is quite correct to omit testing for the end of a DO loop in that case.

4) Segment FR2A in permanent program branches at one point to a cue in overlay (1/4). Investigation has shown however that this branch is never executed in practice unless overlay (1/4) is already in store.

5) The routine STLST in segment FL2D calls APERW which is in (1/1). However STLST is used only by overlay (1/1) so the correct overlay must be in store.

6) Segment FU2A branches to overlay (1/2). However the correct overlay is necessarily in store when this is done since the marker EMARK is tested first.

8. Approximate sizes of the overlays at time of writing

	PROGRAM	LP and LT
(1/1)	178	42
(1/2)	384	148
(1/3)	1699	62
(1/4)	2131	146

Hence : Overlaid LP area = 148 words.

Overlaid Program area = 2131 words.

9. Other modifications introduced in the 10K version

1) When reading a data subfile from EDS, some checks are made on the format of the input. If the bucket header indicates that there are more than 128 words in the bucket, an error 44 is flagged and the compiler assumes that there are 128 words to be dealt with. If a record has a word count of zero or a number greater than 21, an error 43 is flagged and the rest of the bucket is ignored. If a record has a word count of 1, the record is interpreted as a blank line and no error is flagged.

2) It is now possible to specify a subfile generation number in a READ FROM (MT, ...) statement.

3) If the file name in a READ FROM (MT, ...) statement corresponds with that of a tape which is already allotted, the tape is rewound, rather than released and re-allotted.

4) The compiler will work with the new double-buffered paper tape and card output routines.

5) The processing of the EXTERNAL statement has been re-written as in the magnetic tape compiler.

P.M. Girard.