PLAN REFERENCE MANUAL PART 3

Recording, Compiling and Testing Programs
Chapter 7 Compilers

CONVERSION OF SOURCE PROGRAM TO OBJECT PROGRAM

After a PLAN source program has been written, its segments are punched into cards or paper tape as a direct transcription, in character form, of the manuscript version. When the program is in the computer's store ready for execution, it is in a binary form.

The conversion from the external form to the internal is performed on the computer in three logically distinct phases:

1. Compilation: each segment is individually translated to a more basic form known as 'semi-compiled' form.

2. Consolidation: the semi-compiled segments, together with any subroutines required from the library, are merged together to make up the complete program, but the resultant program is still in semi-compiled form.

3. Loading: the complete program, after consolidation, is read (loaded) into the store, this phase including the final translation from semi-compiled form to the binary (machine-language) form.

The same computer or different computers can be used for each of these phases. If all segments of a program, including any library subroutines needed, are presented for compilation at the same time, compilation and consolidation can be done as a single run on the computer, though the logical distinction between them remains. Each of the three phases is described in separate sections below.

Phase 1: Compilation

During compilation from PLAN source language to the semi-compiled form, the PLAN compiler regards each source program statement as specifying information relating to one or another of five categories associated with the division of the object program's storage space: Lower Variable Data, Lower Preset Data (including Literals), Program Instructions, Upper Preset Data, and Upper Variable Data. (These categories are dealt with more fully in Chapter 2.) Associated with each of these categories is a 'relativizer', these relativizers being denoted by the codes LV, LP, LT, PR, UP and UV respectively.

Whenever the compiler processes a PLAN statement and finds there a new symbolic identifier, whether a data name or a label, it allocates that identifier a position within the appropriate category and associates with it:

1. A number indicating its position within the category.
2. The appropriate relativizer.

For example, suppose that the first (or only) statements defining lower variable data are:

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>SUB</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>OPERAND</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>#LOWER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NETT, TAX, BONUS, CARD (20), OUT (31)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Then the compiler would associate with each of these data names the relativizer LV and would assign position numbers to them as follows:

- NETT: 0
- TAX: 1
- BONUS: 2
- CARD: 3
- OUT: 23
Statements concerning upper variable data will be treated in a similar way, the only difference being that the relativizer will be UV instead of LV.

A similar procedure is also adopted for the evaluation of labels whether of constants or program instructions. The label is associated with the appropriate relativizer, which may be a program or an upper or lower relativizer, and the position of the label with regard to that relativizer is determined by the number of words occupied by preceding constants or instructions in the same category of store.

In the program instructions, there may be several program instructions before the first labelled one, and the position number of the first label under PR is therefore not necessarily 0.

Suppose, for example, part of a program reads as follows:

| CARD | OPERAND | ALG | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | NUM | DECIMAL |
| BLOWER | NETT(1), CARD(2.0), | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BLOWER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MAX | 7, 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B PROGRAM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| # ENTRY | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | LDX | 4, CARD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | LDX | 5, MAX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | TEST | BZL | 6, START | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | NONZ | MOV | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | START | LDX | 6, CARD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Labels would be assigned relativizers and position numbers as follows:

- LP
- MAX 0
- PR
- TEST 2
- NONZ 3
- START 4

All the program symbols, together with their position numbers within store categories, are stored in the compiler's 'Symbol Table'. The compiler maintains several tables to assist in evaluating data and instructions. As well as the symbol table there are the 'Branch Ahead', 'Literal' and 'Macro-Instruction' tables and the cue list. The cue list is the master table used for evaluating cues and the store category relativizers themselves. The literal table contains a list of all literal values, indexed to the appropriate instruction operands. The function of the branch ahead table is explained below.

The compiler also counts up the total number of store locations required by the segment for each category.

For constants and program instructions, the compiler must generate not only the information described above to allocate storage, but also the actual quantity that is to be stored. For present purposes, these can be thought of as being of three types:

1. Those which either do not quote any symbolic identifiers, or use only symbolic identifiers that have been defined by a #DEFINE or #SET directive using an absolute expression.
2. Those quoting data names or labels that have previously been set.
3. Those quoting labels that have yet to be defined in the source program (which only applies to the operand field in branch instructions or the modifier part of an index word.)

When the compiler reads a constant of the first type, it will generate the binary form and insert, if required, the value of any previously defined symbol. The result is then output as part of the semi-compiled program.

To illustrate the treatment of the second type, suppose that the instruction

LDX 6 OUT(1)

appears (either as a constant or as a program instruction) and that, as in the earlier example, OUT has already been encountered and assigned position number 23 in the lower variable (LV) category. Then the compiler will generate an instruction in binary form with the operation field set to seven-bit code corresponding to LDX, the three-bit accumulator field set to value 6 (110), the two-bit modifier field
set to value 1 (01) and the twelve-bit N-address field set to value 23 (i.e. the position number of OUT). (Stored instruction statement formats are described in Chapter 1.)

In semi-compiled form, each statement consists of five characters. The last four characters represent the 24-bit instruction generated as described above. The fifth character represents a special code indicating what further operation must be performed on the N-address field. In the illustration above, this code will indicate that the value of the relativizer LV must be added to the value 23 in the N-address field, thereby converting it to an absolute program address. (Note that the value of LV will not be known until the end of consolidation.)

An essentially similar process would be followed if the PLAN statement being compiled were, say, an index word in which the modifier part was a data name.

For a statement of type 3, a branch instruction referring to a label yet to be defined, the name in the operand field of the instruction is entered in the compiler symbol table (unless it is already present, having appeared in a previous operand) and has associated with it the relativizer BA and a number referencing it to an entry in the branch ahead table.

An entry is then made in the branch ahead table to record the relative address and store category at which the instruction occurred, and the branch ahead number of the undefined operand. There will be one entry in this table for every branch instruction or index word that has an undefined operand.

The compiler then, as for other instructions, converts the instruction to binary as far as possible and outputs the result, together with indications of what further operations are required.

When the undefined label is eventually encountered in the label field in the program instruction area, the compiler outputs a series of instructions that will eventually cause the loader to go back to the operand of the appropriate instructions and put in the correct address. When an instruction has been completed in this way, the associated entry is removed from the table.

At the end of each segment of the program the compiler will print out a list of the branches ahead in the table for which no label has been found. At the end of the program if there are any entries remaining in the branch ahead table for which there are no labels, then, during the consolidation phase, they are dealt with in one of the following ways:

1. PLAN compilers other than disc compilers: a very high address, #75777, is allocated to the instruction operands involved.

2. PLAN compilers with disc output: a word containing a SUBWT 2HQQ instruction is inserted in the object program, as the last word of the lower preset data store, and the address of this word is allocated to the instruction operands involved.

3. In a PLAN 4 program, if the "error segment" facility (see under #ERRORSEG and #PROGRAM in Chapter 6) is used, the address of the specified "error segment" is allocated to the instruction operands involved; otherwise, they are treated as in 1 and 2 above.

In each of these cases a warning message is output on the console typewriter.

If a listing during compilation has been called for then as each constant or instruction is converted, it is printed in both source language form and a form related to its semi-compiled format (see Chapter 9 for a full description of compiler listings).

Thus a constant which could be completely converted would be printed first exactly as it appeared in the PLAN program and then as the octal representation of its binary form. For example, if the source program included:

```
#LOWER
CONST #41,3/0,0.5
```

then the listing would include:

```
#LOWER
CONST #41,3/0,0.5
#00000041
#00300000
#20000000
```
Similarly, the instruction discussed earlier:

LDX 6 OUT(1)

would appear on the listing as:

LDX 6 OUT(1) 000 6 1 23 LV

The address OUT is represented on the right-hand side by its position number and relativizer.

Compilation continues, statement by statement, as described above, until the #END directive marking the end of the segment is encountered. When this happens, a 'segment leader' is output at the end of the semi-compiled segment. This leader contains details of:

1. The number of locations in the various categories (e.g. lower variable data, program instructions).
2. Any unsatisfied cues; that is, any names that have appeared in the operand fields of branch instructions but for which no matching labels have been found in the segment.
3. Any cue names defined in this segment by #CUE directives.
4. The names of any common data areas within the segment.
5. Peripherals requested under #PERIPHERAL directives.

Phase 2: Consolidation

The purpose of the consolidation phase is to merge together the constituent segments (including library subroutines) of a program. There is no change from the semi-compiled form of the constants and instructions.

The merging is achieved by taking, from the individual segment leaders, the number of locations in the various categories for each segment and computing from them the initial settings of the relativizers for the program as a whole, and by associating names of common data areas and relating together cue names with unsatisfied cues.

Consider first a single-segment program which is complete in itself and therefore requires no library subroutines. Then the segment leader will carry details of the length of each of the categories, for example:

- Lower variable - 1500 words
- Lower preset - 50 words
- Literals - 10 words
- Program instructions - 450 words
- Upper preset - 200 words
- Upper variable - 1000 words

The consolidator will take these figures and compute actual values for the relativizers. It will start by setting relativizer LV (lower variable) to 48, the standard value for lower variable data when the subprogramming facilities are not in use. Then the other relativizers will be computed as follows:

LP = LV + 1500 = 1545
LT = LP + 50 = 1595
PR = LT + 10 = 1605
UP = PR + 450 = 2055
UV = UP + 200 = 2255

The consolidator will also compute the total store requirement for the program as UV + 1000 = 3255 words.

It will be realized that these values for the relativizer are in fact the starting addresses in store, relative to word 0 of the program, for the various categories of information.

Having done this, the consolidator will output:
A copy of the semi-compiled program (this is optional).

A request slip containing:
(a) the program name,
(b) the program's priority number,
(c) the number of store locations needed,
(d) a list of the slow peripherals reserved by the #PERIPHERAL directive.

A General Purpose Loader (see 'Phase 3: Loading' below).

A consolidated leader, giving the actual values for the various relativizers.

Consolidation of a multi-segment program follows essentially the same lines, with the difference that, when computing the values of the relativizers, allowance is made for appropriate parts of preceding segments.

Consider a two-segment program as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>Segment A</th>
<th>Segment B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower variable</td>
<td>1500</td>
<td>600</td>
</tr>
<tr>
<td>Lower preset</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>Literals</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Instructions</td>
<td>450</td>
<td>350</td>
</tr>
<tr>
<td>Upper preset</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Upper variable</td>
<td>1000</td>
<td>2500</td>
</tr>
</tbody>
</table>

Then the relativizers will be computed as follows:

\[
\begin{align*}
LV & = 45 \\
LP &= LV + 1500 + 600 = 2145 \\
LT &= LP + 50 + 200 = 2395 \\
PR &= LT + 10 + 20 = 2425 \\
UP &= PR + 450 + 350 = 3225 \\
UV &= UP + 200 + 100 = 3525
\end{align*}
\]

and the total storage requirement will be computed as \( UV + 1000 + 2500 = 7025 \).

The consolidator will also examine the list of unsatisfied cues within each segment and endeavour to satisfy them from the cue names of other segments. If there are any that cannot be satisfied in this way, then (for those compilers whose libraries are on magnetic tape) the consolidator will search the library tape, on the assumption that they are calls for library subroutines. Any which cannot be satisfied from the library tape are then reported on the console typewriter. Items gathered from the library tape are in the same semi-compiled form as the PLAN segments, and their storage requirements are allowed for in computing the values of all the relativizers.

Finally, the consolidator will output

1 the semi-compiled program,
2 the request slip,
3 the General Purpose Loader,
4 the consolidated leader.

There are various points that should be noted about the consolidation phase. Firstly, all segments presented up to the reading of #FINISH will be consolidated, regardless of whether or not they specify the same (or any) program name. The name given to the consolidated program will be that specified in the first segment presented to the consolidator having a program name present.
During the search of the library tape, if one is instituted, the consolidator works in a different mode. It will accept for consolidation all segments or subroutines from the appropriate subroutine block on the library tape which meet one or other of the following requirements:

1. if they satisfy one of the program's unsatisfied cues, or
2. if they have a program name present.

It follows from the preceding points that if a user wishes to incorporate free-standing segments into the library tape, he should in general ensure that such segments do not have a program name present; for if they did, they would be incorporated into all programs for which the consolidator searched the particular subroutine block.

Subroutines accepted from the library tape may themselves call other subroutines, and thus give rise to additional cues which must be satisfied by subroutines further down the library tape.

**Phase 3: Loading**

In this final phase, the consolidated, semi-compiled program is converted into binary form by loading it into the computer's store. Loading is effected partly by Executive and partly by the General Purpose Loader, and is initiated by the operator.

The operator types a load message such as:

```
LOAD #ABCD ON 2
```

where #ABCD represents the program's name and 2 typifies the absolute number of a peripheral device from which the consolidated program will be read.

If the program is on magnetic tape, cassette tape or E.D.S., a suitable form of the FIND message is used instead of LOAD. On processors without a control typewriter, the operator loads the program by means of messages input via switches.

The operator action causes Executive to read the request slip, to allocate the storage space and peripherals required (if available), and to enter the program in its list of programs.

Next, Executive reads the General Purpose Loader into that region of the store which Executive has allocated to the program. The General Purpose Loader (G.P.L.) is a program in binary form, which is automatically output with every consolidated program.

Executive then transfers control to the G.P.L., which reads the constants and instructions of the consolidated program. These are still in the semi-compiled form, and the G.P.L. converts them to binary form and stores them. For this, the G.P.L. uses the extra characters in the semi-compiled form to determine which relativizers are to be added to the four characters which represent the quantity. As the binary values are created, they are stored initially beyond the end of the G.P.L. At this time, there is no store explicitly allocated for variable data (lower or upper).

When all the constants and instructions have been converted and stored, the G.P.L. then overwrites itself with zeros, except for a special section which is held in some of the reserved locations of the program's area. This remaining section of the G.P.L. then moves the program word by word to bring the constants and instructions into their correct positions, that is, with the first lower preset word immediately following the last location required for lower variable data.

One special point arises in connection with programs that need little store for variable data, namely that the amount of store actually allocated for lower and upper variables combined cannot be less than the amount occupied by the G.P.L. plus the cue list. This is because the loading operation requires an area of store at least as large as the G.P.L. plus the cue list plus the constants and instructions of the program.

As described above, the sequence in which the consolidated program is read when loading is: (1) request slip, (2) G.P.L., (3) consolidated leader, (4) semi-compiled program; although the semi-compiled program is actually produced before the other items.
COMPILERS AND FACILITIES

There are various PLAN compilers available. The compiler that is used in any particular instance depends upon the size of the central processor, the configuration available and the media being used, as well as the version of PLAN concerned. Details of all the PLAN compilers are given in the table on the next page. Fuller details of PLAN 3, PLAN 2 and PLAN 1 compilers are in the separate sections following, PLAN 4 compilers, which provide additional facilities including the compilation of extended data mode and extended branch mode programs, are described separately in Chapter 8.

Throughout the remainder of this chapter the term 'exchangeable disc' should be taken to include twin exchangeable disc stores.
# The PLAN Compilers

<table>
<thead>
<tr>
<th>Environment</th>
<th>Compiler</th>
<th>Language</th>
<th>Paper Tape Reader or Card Reader</th>
<th>Line Printer</th>
<th>Paper Punch or Card Punch</th>
<th>Cassette Tapes</th>
<th>Magnetic Tapes</th>
<th>Excik Disk Files</th>
<th>Fixed Disk Files</th>
<th>Page Reference in this Chapter or Chapter Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K Processor</td>
<td>XLP</td>
<td>PLAN 1</td>
<td>1</td>
<td>Opt</td>
<td>Opt</td>
<td>3</td>
<td></td>
<td></td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>With basic peripherals only</td>
<td>XLPQ</td>
<td>PLAN 1</td>
<td>1</td>
<td>Opt</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>With cassette tapes</td>
<td>XPLD</td>
<td>PLAN 2</td>
<td>1</td>
<td>Opt</td>
<td>Opt</td>
<td>3</td>
<td></td>
<td></td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>With basic peripherals only</td>
<td>XPLR</td>
<td>PLAN 3</td>
<td>Opt</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>With cassette tapes</td>
<td>XPLS</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>With magnetic tapes</td>
<td>XPLH</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>With disc stores</td>
<td>XPLW</td>
<td>PLAN 3</td>
<td>Opt</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>With magnetic tapes</td>
<td>XPLM</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>With disc stores and cassette tapes</td>
<td>XPLX</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>16K Processor</td>
<td>XPLE</td>
<td>PLAN 3</td>
<td>1</td>
<td>Opt</td>
<td>Opt</td>
<td>3</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>With basic peripherals only</td>
<td>XPLG</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>With magnetic tapes</td>
<td>XPLV</td>
<td>PLAN 3</td>
<td>Opt</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>With disc stores</td>
<td>XPLF</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>With magnetic tapes and disc stores</td>
<td>XPLZ</td>
<td>PLAN 3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>32K Processor</td>
<td>XPLY</td>
<td>PLAN 3</td>
<td>Opt</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>With magnetic tapes</td>
<td>XPLN</td>
<td>PLAN 4</td>
<td>1</td>
<td>1</td>
<td>3+</td>
<td></td>
<td></td>
<td>Chap. 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With disc stores</td>
<td>XPLT</td>
<td>PLAN 4</td>
<td>1</td>
<td>1</td>
<td>2+</td>
<td></td>
<td>Chap. 8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes on the table:**

1. Optional basic peripherals are indicated by "Opt". Optional input devices are optional only in the sense that they are not required if the input tape is called COMPIL FILE (see page 21 of this chapter, and page 34 of Chapter 10). The usage of optional basic peripherals for output depends on whether compiler listings and/or object program output are required. For magnetic device usage, object program output is assumed; fewer units are required if compiler listings only are to be produced. The program library is included if it is retained for reading in overlays, but is not included for non-overlaid compilers.

2. Page reference: for PLAN 3 compilers see also the general information on pages 9 to 10 of this chapter.
PLAN 3 COMPILERS

PLAN 3 compilers provide full facilities for the compilation of compact mode programs. In addition to all the normal features of the PLAN 3 language, they can generate calling sequences to the following software packages:

- MONITOR.
- Input/Output Generator.
- Magnetic Tape Housekeeping system.
- Storage Device Housekeeping systems (15-bit address mode versions).
- Dump and Restart package.
- Overlay system (all except #XPLE, which has basic peripheral output).

All PLAN 3 compilers as supplied have a priority of 93. The number of words of store used by each PLAN 3 compiler is given in Appendix 7. All PLAN 3 compilers except #XPLE are overlay programs; all except #XPLE are batch compilers.

PLAN 3 compilers fall into six major groups:
1. Compilers with basic peripheral input and basic peripheral output.
2. Compilers with basic peripheral input and magnetic tape or cassette tape output.
3. Compilers with input and output on magnetic tape or cassette tape.
4. Compilers with basic peripheral or magnetic tape input and magnetic tape output.
5. Compilers with basic peripheral input and disc output.
6. Compilers with disc input and disc output.

The compilers which fall into each of these groups are shown in the accompanying table.

<table>
<thead>
<tr>
<th>Group</th>
<th>Input on</th>
<th>Output on</th>
<th>Compiler</th>
<th>Description on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Paper tape or cards</td>
<td>Paper tape or cards</td>
<td>XPLE</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Paper tape or cards</td>
<td>Cassette tape</td>
<td>XPLS</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Paper tape or cards</td>
<td>Magnetic tape</td>
<td>XPLG</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XPLH</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XPLL</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Cassette tape</td>
<td>Cassette tape</td>
<td>XPLR</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Magnetic tape</td>
<td>Magnetic tape</td>
<td>XPLV</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XPLW</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Paper tape or cards or magnetic tape</td>
<td>Magnetic tape</td>
<td>XPLY</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Paper tape or cards</td>
<td>Exchangeable disc</td>
<td>XPLM</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Paper tape or cards</td>
<td>Exchangeable disc or fixed disc</td>
<td>XPLF</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Exchangeable disc</td>
<td>Exchangeable disc</td>
<td>XPLX</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Exchangeable disc or fixed disc</td>
<td>Exchangeable disc or fixed disc</td>
<td>XPLZ</td>
<td>18</td>
</tr>
</tbody>
</table>

Details of the individual compilers within each of these groups are given in the following sections of this chapter. These in turn are followed by sections on the internal tables which are set up by the compilers; on the parameters which may be used to specify input and output files and successor programs; and on compiler steering lines. Operating instructions for the PLAN 3 compilers are given on pages 40 to 42, and exception conditions are then dealt with separately for each group of compilers.
Magnetic Tape and Cassette Tape Usage with PLAN 3 Compilers

These standards for magnetic tape and cassette tape usage apply for all PLAN 3 compilers using either of these media.

Any tape which is intended to be opened as a compiler work tape must be specifically labelled SCRATCH TAPE. Unless specified otherwise by a compiler parameter, any tape which is to be opened as a compiler output tape must be specifically labelled SCRATCH TAPE.

Any tape which is opened to receive consolidated semi-compiled object program output is relabelled PROGRAM TAPE (PROGRAM XPKP if cassette tape), with a file generation number of 0 and a retention period of 4095 days, unless specified otherwise by compiler parameters. Any tape which is opened to receive unconsolidated semi-compiled output is relabelled SEMICOMPTAPE, with a file generation number of 0 and a retention period of 4095 days, unless specified otherwise by compiler parameters.

Compiler parameters, which permit the user to specify input and output tapes and to relabel output tapes with a file name, generation number and retention period other than as detailed above, are described in the section entitled ‘PLAN 3 Compiler Parameters’ commencing on page 21.

Output tapes are written in 1900 series standard subfile format. Each subfile is named in accordance with the program name contained on the first #PROGRAM title line of that program encountered by the compiler, in the form ‘PROGRAM NAME’ on magnetic tape or ‘NAME’ on cassette tape. If that line has no program name present, then the subfile is named ‘PROGRAM XXXX’ if on magnetic tape or ‘XXXX’ if on cassette tape.

Work tapes are left scratch at the end of the run. Work tapes and output tapes are closed at the end of the run. Input tapes are unloaded at the end of the run.

Object Program Names

During compilation, the name by which a program is known to the compiler is the program name contained on the #PROGRAM title line of the first segment compiled (whether for the production of object program, or only for the production of a compiler listing). If that line has no program name present, then the name #XXXX is used by the compiler whenever the latter refers to the program in an output message.

When a program is consolidated, whether by a consolidator integral with the compiler or by a separate consolidator program, it is given the name which was contained on the #PROGRAM title line of the first segment to be passed forward for consolidation; that is, the first segment of the program which had OBJECT (or a facility which implies OBJECT) in its steering information. (For an explanation of compiler steering see pages 34 to 39.) If the first segment encountered for consolidation did not have a program name on its #PROGRAM title line, then the consolidator gives the program the name #XXXX.

It follows that if the first segment input to the compiler has LIST or SHORTLIST in its steering information, but does not have OBJECT (or a facility which implies OBJECT), then the name given to the object program may differ from the program name appearing on console messages and line printer messages output during compilation.

PLAN 3 COMPILERS WITH BASIC PERIPHERAL INPUT AND OUTPUT

The PLAN 3 Compiler #XPLE

The only compiler in this group is #XPLE, which is designed for 16K processors.

Form of Source Input

The overall unit of compilation is one source program punched on eight-track paper tape or 80-column cards. A program consists of one or more segments headed by a steering line and terminated by a #FINISH directive. A segment is the basic unit of compilation and consists of PLAN statements headed by a #PROGRAM title line and terminated by a #END directive.

Object Program Output

If object program output is requested by steering information, unconsolidated semi-compiled object program is output to paper tape or cards, as determined by the setting of switch 22 (see ‘Basic Peripheral Usage’ below).

To obtain consolidated semi-compiled program a separate consolidation run, using #XPCA for object program on paper tape or #XPCC for object program on cards, is required.
Basic Peripheral Usage

INPUT DEVICE

The input device, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions for PLAN 3 Compilers' on page 40) and is released when either the #FINISH directive is read or there is a change in input medium. The input medium can be changed at any time by a #SWITCH directive. When this is encountered in the input the present reader is released and one of the other type is allocated.

If a paper tape reader is used, it is disengaged after reading the steering line and after each #END directive in the source program.

PAPER TAPE PUNCH

If switch 22 is off, the compiler allocates a paper tape punch as TP0 when a program with OBJECT on its steering line is presented, and releases it when the #FINISH directive is read. If switch 22 is on, the output is on cards.

If switch 20 is on, the compiler allocates a paper tape punch as TP1, and releases it when the #FINISH directive is read. TP1 is used for output which would otherwise have been produced on the line printer; a line printer is not allocated in these circumstances.

It is possible, therefore, for two paper tape punches to be allocated.

CARD PUNCH

If switch 22 is on, the compiler allocates a card punch as CP0 when a program with OBJECT on its steering line is presented, and releases it when the #FINISH directive is read. If switch 22 is off, the output is on paper tape.

LINE PRINTER

Unless switch 20 is on, a line printer is allocated as LP0 when the compiler is entered, and is released at the end of the run. In addition to providing a compiler listing if so requested by steering information, the line printer is used for certain error messages (see 'Exception Conditions', page 42).

If switch 20 is on, no line printer is allocated, a tape punch being allocated in its place.

PLAN 3 COMPILERS WITH BASIC PERIPHERAL INPUT AND MAGNETIC TAPE OR CASSETTE TAPE OUTPUT

The PLAN 3 Compilers #XPLG, #XPLH, #XPLL, #XPLS

There are four compilers in this group. Of the magnetic tape output compilers, #XPLG is designed for 16K processors and #XPLH is designed for 8K processors. #XPLL is an alternative compiler for 8K processors which utilizes exchangeable disc files to increase the speed of compilation. The cassette tape output compiler is #XPLS, designed for an 8K processor.

Form of Source Input

The overall unit of compilation is a batch of source programs punched on eight-track paper tape of 80-column cards. A batch consists of one or more programs, the last of which is terminated by a #STOP directive. A program consists of one or more segments headed by a steering line and terminated by a #FINISH directive. A segment is the basic unit of compilation and consists of PLAN statements headed by the #PROGRAM title line and terminated by a #END directive. A segment may be preceded by a segment level steering line (except with #XPLS).

The batch may optionally be preceded by parameters, as described on page 21.

This paragraph applies to #XPLG, #XPLH, and #XPLL only. If the #END directive is omitted from the last segment of a program, the #FINISH directive is taken to imply a #END. If the #FINISH directive is omitted from the last program of a batch, the #STOP directive is taken to imply a #FINISH. If neither the #END directive nor the #FINISH directive precedes the #STOP directive, the #STOP directive is taken to imply both these missing directives. The implied directives will be printed and flagged as an 'R' error in any compiler listing of the segment concerned.

Object Program Output

If object program output is requested by steering information, consolidated semi-compiled object programs are written to a magnetic tape (#XPLS, cassette tape) in accordance with the standards for tape usage described on page 10.
Basic Peripheral Usage

INPUT DEVICE
The input device, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions for PLAN 3 Compilers', page 40) and is released when either the #STOP directive is read or there is a change of input medium. The input medium can be changed at any time after the parameters, if any, have been read, by a #SWITCH directive. When this is encountered in the input the present reader is released and one of the other type is allocated.

If a paper tape reader is used, it is disengaged after reading each program level steering line, after each #END directive and after each #FINISH directive.

LINE PRINTER
A line printer is allocated as LPO when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings or store maps that may be requested by steering information, the line printer is used for certain error messages (see 'Exception Conditions', page 43 for the magnetic tape compilers and page 46 for #XPLS) and to print any of the optional parameters which may precede the source programs.

Magnetic Tape Usage, #XPLG
Magnetic tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY MAGNETIC TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MT0 until the compiler is deleted.

OPTIONAL MAGNETIC TAPES
If all the programs in the batch being compiled are only to be listed, no further magnetic tapes are required.
If a program or segment with OBJECT (or a facility which implies OBJECT) on its steering line is presented, the compiler opens a work tape as MT2 and an output tape as MT3.

Magnetic Tape Usage, #XPLH
Magnetic tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY MAGNETIC TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MT0 until the compiler is deleted.

A magnetic tape is opened as MT3. This tape is relabelled as for an output tape, but is used initially as a work tape. If any program or segment in the batch being compiled has OBJECT (or a facility which implies OBJECT) on its steering line, MT3 subsequently receives the compiled output. If all the programs in the batch are only to be listed, MT3 is scratched at the end of the run.

OPTIONAL MAGNETIC TAPE
If all the programs in the batch are only to be listed, no further magnetic tapes are required.
If a program or segment with OBJECT (or a facility which implies OBJECT) on its steering line is presented, the compiler opens another work tape as MT2.

Magnetic Device Usage, #XPLL
Magnetic tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY MAGNETIC TAPE
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MT0 until the compiler is deleted.

OBLIGATORY EXCHANGEABLE DISC FILES
When the compiler is entered an exchangeable disc scratch file of 80 one block buckets is allocated as ED1. A
second exchangeable disc scratch file of 240 one block buckets is allocated as ED0, and the compiler’s overlays are
copied to this file from MT0. The compiler thereafter overlays itself from ED0 instead of from the P.L.T., the time
taken in transcription being more than offset by the greater speed of overlaying from exchangeable disc. The P.L.T.
is retained.

OPTIONAL EXCHANGEABLE DISC FILE
If all the programs in the batch being compiled are only to be listed, no further exchangeable disc files are required.
If a program or segment with OBJECT (or a facility which implies OBJECT) on its steering line is presented, a
further exchangeable disc scratch file of 80 one block buckets is opened as ED2.

EXCHANGEABLE DISC FILE EXTENSION
If ED1 or ED2 is filled, the compiler attempts to extend it. If it fails, it releases the overlay scratch file, ED0,
making its space available for use; for the rest of the run, the compiler overlays itself from the P.L.T. If a
subsequent attempt to extend a file fails, diagnostic messages are output (see page 43).

OPTIONAL MAGNETIC TAPE
If all the programs in the batch being compiled are only to be listed, no further magnetic tapes are required.
If a program or segment with OBJECT (or a facility which implies OBJECT) on its steering line is presented, an
output tape is opened as MT3.

Cassette Tape Usage, #XPLS
Cassette tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY CASSETTE TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as CT0 until the compiler is
deleted.
When the compiler is entered, a work tape is opened as CT1.

OPTIONAL CASSETTE TAPES
If all the programs in the batch being compiled are only to be listed, no further cassette tapes are required.
If a program with OBJECT on its steering line is presented, an output tape is opened as CT3 and a further work
tape is opened as CT2.

SETTING UP CONSIDERATIONS
On machines without a console typewriter, the cassette tapes to be used must be mounted on the correspondingly
numbered stations; that is, the tape to be opened as CT1 must be mounted on station 1, and so on.
On machines with a console typewriter, the cassette tapes may be mounted on any station, and it is worth consider-
ing the order in which they should be mounted to obtain the most efficient run. This depends on the order in
which they are opened, which in turn depends on the steering information present. The various cases are considered
below, assuming that the P.L.T. is already mounted on station 0:

1  No program in the batch has OBJECT or BINARY on its steering line.
   CT1 is opened. CT2 and CT3 are not required.
   CT1 should be mounted on the lowest-numbered available station.

2  The first program in the batch has OBJECT or BINARY on its steering line.
   CT3 is opened first, then CT2, then CT1.
   CT3 should be mounted on the lowest-numbered available station, CT2 on the next lowest, CT1 on
   the next lowest.
   On a single cassette tape unit, the least efficient setting up arrangement (CT1, CT2, CT3 on stations
   1, 2 and 3 respectively) would be three alignments slower.

3  The first program in the batch has neither OBJECT nor BINARY on its steering line. A later program in the
   batch has OBJECT or BINARY on its steering line.
CT1 is opened for the first program.
CT3, then CT2, are opened for the later program.
CT1 should be mounted on the lowest-numbered available station, CT3 on the next lowest, CT2 on the next lowest.
On a single cassette tape unit, the least efficient setting up arrangement (CT2, CT3, CT1 on stations 1, 2 and 3 respectively) would be three alignments slower.

PLAN 3 COMPILERS WITH INPUT AND OUTPUT ON MAGNETIC TAPE OR CASSETTE TAPE

The PLAN 3 Compilers #XPLV, #XPLW, #XPLR

There are three compilers in this group. Of the magnetic tape compilers, #XPLV is designed for 16K processors and #XPLW is designed for 8K processors. The cassette tape compiler is #XPLR, designed for an 8K processor.

Form of Source Input

The overall unit of compilation is a magnetic tape (#XPLR, cassette tape) in 1900 series standard subfile format, written by a COSY compilation system editor program. These editor programs (#XPMR and #XPM5 for magnetic tape, and #XPMJ and #XPMK for cassette tape) are fully described in Chapter 10. The tape contains one or more source and/or semi-compiled programs.

A source program consists of one or more segments each headed by a record containing steering information. A segment is the basic unit of compilation and consists of PLAN statements headed by the steering information and #PROGRAM title line, and terminated by a #END directive. Semi-compiled segments may also be input for consolidation from the same input tape.

Object Program Output

#XPLV, #XPLW

If object program output is requested by steering information, consolidated semi-compiled object programs are written to magnetic tape in accordance with the standards for tape usage described on page 10.

#XPLR

If object program output is requested by steering information, unconsolidated semi-compiled object programs are written to cassette tape in accordance with the standards for tape usage described on page 10.

To obtain consolidated semi-compiled program a separate consolidation run, using #XPCX, is required; segments must be in the order required by #XPCX.

Basic Peripheral Usage

INPUT DEVICE

If entry point 22 is used, the compiler opens a tape called COMPILE FILE as the input tape, and no basic peripheral input device is allocated. If the input tape is not called COMPILE FILE, then a different entry point must be used, and a paper tape reader or a card reader is required to read in parameters specifying the input file and optionally the tape to be used for output. These parameters are described in the section entitled 'PLAN 3 Compiler Parameters' commencing on page 21.

If entry point 20 is used a paper tape reader is allocated. If entry point 21 is used a card reader is allocated. The reader is released when the parameter terminator is read.

LINE PRINTER

A line printer is allocated as LP0 when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings or store maps that may be requested by steering information, the line printer is used for certain error messages (magnetic tape compilers only; see 'Exception Conditions', page 43) and to print the compiler parameters.

Magnetic Tape Usage, #XPLV

Magnetic tapes are treated in accordance with the standards for tape usage described on page 10.
OBLIGATORY MAGNETIC TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MT0 until the compiler is deleted.
The input tape is opened as MT4.

OPTIONAL MAGNETIC TAPES
If all the programs in the batch being compiled are only to be listed, no further magnetic tapes are required.
If a segment with OBJECT (or a facility which implies OBJECT) in its steering record is presented, the compiler opens a work tape as MT2 and an output tape as MT3.

Magnetic Tape Usage, #XPLW
Magnetic tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY MAGNETIC TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MT0 until the compiler is deleted.
The input tape is opened as MT4.
A magnetic tape is opened as MT3. This tape is relabelled as for an output tape, but is used initially as a work tape.
If any segment in the batch of programs being compiled has OBJECT (or a facility which implies OBJECT) in its steering record, MT3 subsequently receives the compiled output. If all the programs in the batch are only to be listed MT3 is scratched at the end of the run.

OPTIONAL MAGNETIC TAPE
If all the programs in the batch being compiled are only to be listed, no further magnetic tapes are required.
If a segment with OBJECT (or a facility which implies OBJECT) in its steering record is presented, the compiler opens another work tape as MT2.

Cassette Tape Usage, #XPLR
Cassette tapes are treated in accordance with the standards for tape usage described on page 10.

OBLIGATORY CASSETTE TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as CT0 until the compiler is deleted.
The input tape is opened as CT3.
A work tape is opened as CT1.

OPTIONAL CASSETTE TAPE
If all the programs in the batch being compiled are only to be listed, no further cassette tapes are required.
If a segment with OBJECT in its steering record is presented, an output tape is opened as CT2.

SETTING UP CONSIDERATIONS
On machines without a console typewriter, the cassette tapes to be used must be mounted on the correspondingly-numbered stations; that is, the tape to be opened as CT1 must be mounted on station 1, and so on.
On machines with a console typewriter, the cassette tapes may be mounted on any station. For most efficient running, however, they should be mounted with CT1 on the lowest-numbered available station, CT3 on the next lowest and CT2 (if required) on the next lowest, as the tapes are opened in that order. Other arrangements would be up to three realignments slower.
PLAN 3 COMPILERS WITH BASIC PERIPHERAL OR MAGNETIC TAPE INPUT AND MAGNETIC TAPE OUTPUT

The PLAN 3 Compiler #XPLY

The only compiler in this group is #XPLY, which is designed for 16K processors.

Form of Source Input

The overall unit of compilation is a batch of source programs punched on eight-track paper tape or 80-column cards or a magnetic tape in 1900 Series standard subfile format, written by a COSY compilation system editor program. These editor programs (#XPMR and #XPMS) are fully described in Chapter 10. The tape contains one or more source and/or semi-compiled programs.

A source program on magnetic tape consists of one or more segments each headed by a record containing steering information. A segment is the basic unit of compilation and consists of PLAN statements headed by the steering information and #PROGRAM title line, and terminated by a #END directive. Semi-compiled segments may also be input for consolidation from the same input tape.

A batch on paper tape or cards consists of one or more programs, the last of which is terminated by a #STOP directive. A program consists of one or more segments headed by a steering line and terminated by a #FINISH directive. A segment is the basic unit of compilation and consists of PLAN statements headed by the #PROGRAM title line and terminated by a #END directive. A segment may be preceded by a segment level steering line.

The batch may optionally be preceded by parameters, as described on page

If the #END directive is omitted from the last segment of a program, the #FINISH directive is taken to imply a #END. If the #FINISH directive is omitted from the last program of a batch, the #STOP directive is taken to imply a #FINISH. If neither the #END directive nor the #FINISH directive precedes the #STOP directive, the #STOP directive is taken to imply both these missing directives. The implied directives will be printed and flagged as an "R" error in any compiler listing of the segment concerned.

Note, however, that a batch must be wholly on magnetic tape or on cards or paper tape.

Object Program Output

If object program output is requested by steering information, consolidated semi-compiled object programs are written to magnetic tape in accordance with the standards for tape usage described on page 10.

Basic Peripheral Usage

INPUT DEVICE

The input device, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions for PLAN 3 Compilers', page 40) and is released when either the #STOP directive is read or there is a change of input medium. The input medium can be changed at any time after the parameters, if any, have been read, by a #SWITCH directive. When this is encountered in the input the present reader is released and one of another type is allocated.

If a paper tape reader is used, it is disengaged after reading each program level steering line, after each #END directive and after each #FINISH directive.

If entry point 22 is used, the compiler opens a tape called COMPILE FILE as the input tape, and no basic peripheral input device is allocated. If the input tape is not called COMPILE FILE, then a different entry point must be used, and a paper tape reader or a card reader is required to read in parameters specifying the input file and optionally the tape to be used for output. These parameters are described in the section entitled "PLAN 3 Compiler Parameters" commencing on page 21.

If entry point 20 is used a paper tape reader is allocated. If entry point 21 is used a card reader is allocated. The reader is released when the parameter terminator or a #STOP directive is read.

LINE PRINTER

A line printer is allocated as LPO when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings or store maps that may be requested by steering information, the line printer is used for certain error messages (see 'Exception Conditions', page 43) and to print the compiler parameters.
**Magnetic Device Usage**

Magnetic tapes are treated in accordance with the standards for tape usage described on page 10. The compiler is an overlay program loaded from the P.L.T. or from an exchangeable disc file.

**OBLIGATORY MAGNETIC TAPES**

If the compiler is loaded from the P.L.T., this tape is retained as MT0 until the compiler is deleted.

If the source is on tape, the input tape is opened as MT4.

**OBLIGATORY EXCHANGEABLE DISC FILE**

If the compiler is loaded from an exchangeable disc file, this file is retained as ED0 until the compiler is deleted or if the compiler is loaded from the P.L.T., an exchangeable disc scratch file of 240 one block buckets is allocated as ED0 and the compiler's overlays are copied to this file from MT0. This compiler thereafter overlays itself from ED0 instead of from the P.L.T., the time taken in transcription being more than offset by the greater speed of overlaying from exchangeable disc. The P.L.T. is retained.

**OPTIONAL EXCHANGEABLE DISC FILES**

If all the programs in the batch being compiled are only to be listed, no further exchangeable disc files are required.

If a segment with OBJECT (or a facility which implies OBJECT) in its steering record is presented a further exchangeable disc scratch file of 80 one block buckets is opened as ED2.

If, when object program output is being produced, the compiler encounters a program whose steering segment contains a #OVERLAY directive but does not contain a #ORDER directive, then during the consolidation phase a third exchangeable disc scratch file of 80 one block buckets is opened as ED1.

**EXCHANGEABLE DISC FILE EXTENTION**

If it fails, a diagnostic message is output (see page 44) (if the compiler was loaded from exchangeable disc file) or it releases the overlay scratch file ED0, making its space available for use; for the rest of the run, the compiler overlays itself from the P.L.T. (if the compiler was loaded from the P.L.T.). If a subsequent attempt to extend a file fails, diagnostic messages are output (see page 44).

**OPTIONAL MAGNETIC TAPE**

If all the programs in the batch being compiled are only to be listed no further magnetic tapes are required.

If a segment with OBJECT (or a facility which implies OBJECT) in its steering record is presented, an output tape is opened as MT3.

**PLAN 3 COMPILERS WITH BASIC PERIPHERAL INPUT AND DISC OUTPUT**

**The PLAN 3 Compilers #XPLM, #XPLF**

There are two compilers in this group. #XPLM is designed for 8K processors and has output on exchangeable disc. #XPLF is designed for 16K processors and has output on exchangeable disc or fixed disc.

**Form of Source Input**

The overall unit of compilation is a batch of source programs punched on eight-track paper tape or 80-column cards. A batch consists of one or more programs, the last of which is terminated by a #STOP directive. A program consists of one or more segments headed by a steering line and terminated by a #FINISH directive. A segment is the basic unit of compilation and consists of PLAN statements headed by a #PROGRAM title line and terminated by a #END directive. A segment may be preceded by a segment level steering line.

The program must be preceded by parameters specifying the output file, as described on page 27.

If the #END directive is omitted from the last segment of a program, the #FINISH directive is taken to imply a #END. If the #FINISH directive is omitted from the last program of a batch, the #STOP directive is taken to imply a #FINISH. If neither the #END directive nor the #FINISH directive precedes the #STOP directive, the #STOP directive is taken to imply both these missing directives. The implied directives will be printed and flagged as an 'R' error in any compiler listing of the segment concerned.
Object Program Output

If object program output is requested by steering information, unconsolidated semi-compiled object programs are output to the disc file specified by the compiler parameters. Each program output is written in a subfile named PROGRAM NAME, where NAME is the program name contained on the first #PROGRAM title line of that program encountered by the compiler, or, if no program name is present on that line, the four characters XXXX.

To obtain consolidated object program in binary a separate consolidation run, using #XPCL after #XPLM or #XPCK after #XPLF, is required. Provision is made, by means of optional parameters (see page 30), for the compiler to delete itself and call the consolidator into core at the end of the run. If so specified, the consolidator may then be entered automatically.

Basic Peripheral Usage

INPUT DEVICE

The input device, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions for PLAN 3 Compilers', page 40) and is released when either the #STOP directive is read or there is a change of input medium. The input medium can be changed at any time after the parameters have been read, by a #SWITCH directive. When this is encountered in the input the present reader is released and one of the other type is allocated.

If a paper tape reader is used, it is disengaged after reading each program level steering line, after each #END directive, and after each #FINISH directive.

LINE PRINTER

A line printer is allocated as LP0 when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings that may be requested by steering information, the line printer is used for certain error messages (see 'Exception Conditions', page 48) and to print the compiler parameters.

Exchangeable Disc Usage, #XPLM

OBLIGATORY EXCHANGEABLE DISC FILES

The compiler is an overlay program loaded from an exchangeable disc file, so this file is retained as ED0 until the compiler is deleted.

When the compiler is entered an exchangeable disc scratch file of 80 one block buckets is allocated as ED1.

OPTIONAL EXCHANGEABLE DISC FILES

If all the programs in the batch compiled are only to be listed, no further exchangeable disc files are required.

If a program or segment with OBJECT on its steering line is presented, a permanent exchangeable disc file as specified by the compiler parameters is opened as ED2, to receive the output.

Disc Usage, #XPLF

#XPLF is an overlay program loaded from an exchangeable disc file, so this file is retained as ED0 until the compiler is deleted.

The output medium, exchangeable disc or fixed disc, is determined by the compiler parameters, as described on page 27.

OPTIONAL DISC FILE

If all the programs in the batch being compiled are only to be listed no disc files are required.

If a program or segment with OBJECT on its steering line is presented, a permanent disc file as specified by the compiler parameters is opened as ED2 or FD2, to receive the output.

PLAN 3 COMPILERS WITH DISC INPUT AND DISC OUTPUT

The PLAN 3 Compilers #XPLX, #XPLZ

There are two compilers in this group. #XPLX is designed for 8K processors and has input and output on exchangeable disc. #XPLZ is designed for 16K processors, receives input from either exchangeable disc or fixed disc, and outputs either to exchangeable disc or to fixed disc; the input medium may be the same as or different from the output medium.
#XPLX is intended mainly for use with the sort/merge generator programs #XSEG.

**Form of Source Input**

The overall unit of compilation is an exchangeable disc file (#XPLZ, or a fixed disc file) written by a DISC COSY compilation system editor program, or by the sort/merge generator program #XSEG, or by a user program whose output file conforms to the specification given on pages 47 to 50 of Chapter 11. The DISC COSY editor programs (#XPMX and #XPMY) are described in Chapter 11 and the sort/merge generator program is described in the manual 'Direct Access Sorting'.

The input file contains one or more programs in source and/or semi-compiled form.

A source program consists of one or more segments each headed by a record containing steering information. A segment is the basic unit of compilation and consists of PLAN statements headed by the steering record and #PROGRAM title line, and terminated by a #END directive. Semi-compiled segments may also be copied across to the output file from the same input file. If separate amendments subfiles exist on the input file, the amendments subfiles will be read at the same stage as their related source subfiles and the amendments incorporated in the segment before it is compiled. If amendments subfiles are fragmented, a maximum of 10 fragments per subfile will be accepted; if an amendments subfile having more than 10 fragments is encountered the compiler halts with the console message HALTED: IF and the run must be abandoned.

If the #END directive is absent from an amended source segment, the compiler will imply #END when the end of the amended segment is reached. The implied directive will be printed and flagged as an 'R' error in any compiler listing of the segment. Except where the input file was written by a user program, the circumstance can arise only as a result of an incorrectly specified amendment in an amend-in-situ run of the DISC COSY editor program #XPMZ.

**Object Program Output**

If object program output is requested by steering information, unconsolidated semi-compiled object programs are output to the exchangeable disc file (#XPLZ, or fixed disc file) specified by the compiler parameters. The compiler parameters are described on page 27. Each program output is written in a subfile named PROGRAM NAME, where NAME is the program name contained on the first #PROGRAM title line of that program encountered by the compiler, or, if no program name is present on that line, the four characters XXXX.

To obtain consolidated object program in binary form a separate consolidation run using #XPCL after #XPLX or #XPCK after #XPLZ is required. Provision is made, by means of optional parameters (see page 30), for the compiler to delete itself and call the consolidator into core at the end of the run. If so specified, the consolidator may then be entered automatically.

**Basic Peripheral Usage**

**INPUT DEVICE**

A paper tape reader or a card reader is required to read in parameters specifying the files to be opened (see page 21).

If entry point 20 is used, a paper tape reader is allocated. If entry point 21 is used, a card reader is allocated. The reader is released when the parameter terminator is read.

**LINE PRINTER**

A line printer is allocated as LPO when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings that may be requested by steering information, the line printer is used for certain error messages (see 'Exception Conditions', page 48) and to print the compiler parameters.

**Exchangeable Disc Usage, #XPLX**

**OBLIGATORY EXCHANGEABLE DISC FILES**

The compiler is an overlay program loaded from an exchangeable disc file, so this file is retained as ED0 until the compiler is deleted.

When the compiler is entered, the input file is opened as ED4, and a scratch file of 80 one block buckets is opened as ED1.

**OPTIONAL EXCHANGEABLE DISC FILE**

If all the programs in the batch being compiled are only to be listed, no further exchangeable disc files are required.
If a segment with OBJECT in its steering record is presented, a permanent exchangeable disc file as specified by the compiler parameters is opened as ED2, to receive the output.

**Disc Usage #XPLZ**

#XPLZ is an overlay program loaded from an exchangeable disc file, so this file is retained as EDO until the compiler is deleted.

The medium used, exchangeable disc or fixed disc, is specified separately for input and output by the compiler parameters, as described on page

**OBLIGATORY DISC FILE**
The input file is opened as ED4 or FD4.

**OPTIONAL DISC FILE**
If all the programs in the batch being compiled are only to be listed, no further disc files are required.
If a segment with OBJECT in its steering record is presented, a permanent disc file as specified by the compiler parameters is opened as ED2 or FD2, to receive the output.

**COMPILER TABLE SIZES**
The PLAN 3 compilers use tables during the compilation process to hold details of the symbolic identifiers, macro-instructions, literals, branches ahead and cues that are encountered. The sizes of these tables are limited, but the compiler will, if the need arises, attempt to expand the combined symbol and branch ahead table by increasing the amount of core store allocated to the compiler. If the attempt is unsuccessful the program currently being compiled is abandoned, and the reason for the abandonment is printed on the line printer. (The action of cassette tape compilers in these circumstances is different; see 'Exception Conditions', page 46.) PLAN 3 compilers with integral consolidators will attempt to expand the consolidated cue list in a similar manner if the need arises. A larger symbol and branch ahead table, and with 8K compilers a larger macro-instruction table also, may be provided by specifying a suitable storage parameter in the FIND or LOAD message when the compiler is loaded.

**Combined Symbol and Branch Ahead Table**
The size of the combined symbol and branch ahead table in PLAN 3 compilers designed for 16K processors is 1600 words. The size of this table in PLAN 3 compilers designed for 8K processors is 800 words.

Every symbol occupies two words plus the number of words required to contain its name in character form.
Segment names, peripheral names, common block names and data names within common areas require a further word.

Every branch ahead occupies two words. A further one or two words are required if the forward reference occurs in a common area.

The sizes quoted above for this table are the 'official' sizes which have been allowed in the compilers. The actual sizes of the table are greater than the official sizes, but this may not be a permanent feature; additionally, compilers will attempt to extend their table size if necessary. If the amount of table space used in the course of a compilation exceeds the 'official' table size and LIST, SHORTLIST, MAP or FULLLIST appears in the steering information, then a warning message is output on the segment's error line, to the right of the error code information. The format of this message is:

*TABLE EXCEEDED NNNN WORDS AT LINE XXXX*

where NNNN is 1600 or 800 depending on which compiler is being used and XXXX is the number of the line which caused the 'official' table size to be exceeded. This message is only a warning, in case the program should subsequently be recompiled with a smaller available table space; it does not mean that the compilation is abortive.

**Literal Table**
The size of the literal table in all PLAN 3 compilers is 75 words.

Every literal occupies one word in the table. One extra word is used (and inserted in the object program) if the literal makes reference to a word in a common area. If the table becomes full during compilation and a listing is being produced, the compilation is interrupted and all 75 literals are printed out. Whether the compilation is being listed or not the literal table is zeroized and is ready to be filled again.
Macro-instruction Table

The size of the macro-instruction table in all PLAN 3 compilers is 500 words.

PLAN macro-instructions take up about 250 words. Every macro-instruction takes up approximately three words plus a further three words for each basic instruction that is requires. For example, BXGE takes up nine words: three in itself plus three for TXL and three for BCC. Apart from PLAN macro-instructions, there is room in the table for about 20 to 30 average-sized, user-defined macro-instructions.

Consolidated Cue List

The size of the consolidated cue list in all PLAN 3 compilers with integral consolidators is 1000 words.

Each cue name occupies three words plus the number of words required to contain its name in character form.

Map/Fulllist Table

This table re-uses some of the locations previously used for the combined symbol and branch ahead table and for the consolidated cue list. Information for the preparation of a map and fulllist, if either of those facilities is requested by steering information, is written to magnetic tape during compilation and consolidation, and is read back into core store when consolidation has been completed. If at that time it is found that there is insufficient space available to read in all the necessary information, the map and/or fulllist is abandoned; object program output, however, will already have been achieved, and although the ‘SERV’ message is output on the console (see ‘Operating Instructions for PLAN 3 Compilers’, page 40) the program may be run if required.

PLAN 3 COMPILER PARAMETERS

Certain PLAN 3 compilers require, and others may optionally accept, parameters to be presented on paper tape or cards as the first item of input. These compilers comprise:

1. Compilers with basic peripheral input and output on magnetic tape or cassette tape. These compilers may optionally accept parameters defining the output tape.

2. Compilers with input and output on magnetic tape or cassette tape. These compilers require parameters defining the input tape (except where a special input tape, named COMPILE FILE, is indicated by the choice of entry point) and may optionally accept parameters defining the output tape.

3. Compilers with basic peripheral input and output on disc. These compilers (unless the program being compiled is only to be listed) require parameters defining the output file.

4. Compilers with input and output on disc. These compilers require parameters defining the input file and (unless the program being compiled is only to be listed) the output file.

Compilers in the last two of the above groups which require a work file may optionally accept a parameter defining the file to be used for that purpose.

The formats of these parameters for all the above groups are based on the following generalised form, with individual fields omitted where they are inappropriate to the device or to the input or output status of the parameter, and with any combination of the underlined fields omitted at the user’s option:

TYPE(FILENAME(FGN1=FGN2/RSN.RET).TSN)

The significances of the individual fields are fully explained where each particular parameter is discussed.

In addition to the file definition parameters, compilers with input on magnetic tape or on disc can accept a parameter instructing them to examine one subfile of the input file only, so that for that run they operate as single program compilers instead of as batch compilers. Provided that they have that parameter present (that is, are working as single program compilers), compilers with output on disc may also optionally accept a parameter specifying a successor program to be loaded automatically at the end of the run; and, if the successor program so specified is a suitable consolidator, a further optional parameter may cause it to be entered automatically, to produce a binary program either in core store or output to a specified disc file. The two last-mentioned parameters are not necessary for compilers with magnetic tape output, as these have integral consolidators and, by means of a steering line facility (see pages 35 to 39) can be instructed to produce, via the overlay loader, a single-program binary tape.

The compiler parameters are printed on the line printer at the commencement of the run. With compilers in groups 1 and 3 above, if there are no parameters present the word NONE is printed under the heading PARAMETERS.
The next three subsections described in detail the parameters which may be used with compilers having output on magnetic tape, on cassette tape and on disc respectively. They are followed by a table that summarises which parameters are applicable to each particular compiler.

Each parameter discussed in this section must start at character position one of a paper tape line or column one of a card. Paper tape lines are terminated by a newline character and have a maximum permitted size of 80 internal characters; beta shift characters and delta shift characters other than $, \], ^, *+, =, newline and horizontal tabulation are not accepted. If used, the compiler parameter(s) must be the first item input during a run. The parameters present may be in any order, except that the parameter terminator, if used, must come last.

Parameters for PLAN 3 Compilers with Magnetic Tape Output

OUTPUT FILE PARAMETERS

Any PLAN 3 compiler with magnetic tape output may optionally be presented with an OUT parameter specifying the tape to be used for output. In the absence of an OUT parameter, the first available tape labelled SCRATCH TAPE is opened as the output tape. If an OUT parameter is present, it may optionally be accompanied by a REN parameter specifying the new header label information to be written to the output tape. In the absence of a REN parameter the output tape is relabelled PROGRAM TAPE, with a file generation number of 0 and a retention period of 4095 days; this applies whether the OUT parameter was also absent or not.

INPUT FILE PARAMETERS

If the compiler has input on magnetic tape then, unless entry point 22 is used, an IN parameter is required to specify the input tape. If the IN parameter is omitted the compiler will seek to open a tape labelled COMPIL FILE, with any file generation number and a zero reel sequence number.

If the compiler has input on magnetic tape and entry point 22 is used, then it will open COMPIL FILE as the input tape and will not allocate a reader, so that no parameters can be read. If it is desired to input COMPIL FILE but to pick up a named tape as the output tape and rename it, it will be necessary to use entry point 20 or 21 (see 'Operating Instructions') and to use OUT and REN parameters.

If entry point 20 or 21 is used, a compiler with magnetic tape input may optionally be presented with a SUB parameter, specifying a subfile to be examined for a single program compilation run. This parameter may additionally include steering information which, if present, will override the steering information contained in the specified subfile on the input tape. This facility is known as 'external steering'.

PARAMETER TERMINATOR

If the compiler has basic peripheral input and parameters (OUT alone, or OUT and REN) are present, they may optionally be followed by PEND, the parameter terminator.

Unless entry point 22 is used, a compiler with magnetic tape input requires the parameter terminator, PEND, in order to release the reader and proceed to the next stage of the run.

PARAMETER SPECIFICATIONS

Details of the formats of the individual parameters follow.

1. The IN parameter:

   IN(FILENAME(FGN/RSN),TSN)

   where FILENAME = the existing 12 character file name. If this is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

   FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted or expressed as zero the file generation number on the tape is not checked.

   RSN = the reel sequence number, in decimal, in the range 0 to 511. The reel sequence number will always be checked, but if it is zero it may be omitted from the parameter, in which case the preceding solidus should also be omitted.

   TSN = the tape serial number, in octal, in the range octal 0 to octal 37777777. It may be punched with or without a preceding * or # sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the tape serial number will not be checked.
If FGN is omitted and RSN is present, the solidus must also be present, thus:
IN(FILENAME(RSN),TSN)
If FGN and RSN are both omitted, the related brackets should also be omitted, thus:
IN(FILENAME,TSN)
If all the fields are omitted (#XPLY only), it denotes that the source input follows on the same peripheral type as the parameters. The entry point will therefore be 20 or 21, and the parameter will thus be:
IN
2 The OUT parameter:
OUT(FILENAME(FGN/RSN),TSN)
where the significances of FILENAME, FGN, RSN and TSN are as described for the IN parameter above.
If FGN is omitted and RSN is present, the solidus must also be present, thus:
OUT(FILENAME(/RSN),TSN)
If FGN and RSN are both omitted, the related brackets should also be omitted, thus:
OUT(FILENAME,TSN)

If a specific tape labelled SCRATCH TAPE is required to be opened as the output tape, then it should be declared in an OUT parameter with SCRATCH TAPE as its file name, and a tape serial number.
If it is intended to rely on Executive's search for a tape with an expired retention period to locate an output tape, then it will be necessary to use an OUT parameter with FILENAME written as ####, thus:
OUT(####)
If it is required to pick up a tape of a given serial number and an expired retention period as the output tape, and there is any uncertainty as to the existing file name, then it is permissible to use a parameter of the form:
OUT(####,TSN)
The sequence of events in this case is that Executive locates the lowest numbered deck containing a tape with an expired retention period, and opens the tape, relabelling it SCRATCH TAPE. The compiler program then proceeds to check the tape serial number. Should this not be the required one, the program unloads the tape and the process is repeated with the next lowest numbered deck. If the correct tape cannot be found on any deck, Executive outputs a message asking for it to be loaded. Any tapes opened and subsequently unloaded in this manner have of course had their header labels overwritten: the facility should therefore be used with care.

3 The REN parameter:
RENAME(FILENAME(FGN,RET))
where FILENAME = the 12 character file name to be written to the output tape. If this is punched with less than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted the tape is given a file generation number of zero.

RET = the number of days retention period to be given to the tape, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted the tape is given a retention period of 4095.
If FGN is omitted and RET is present, the preceding comma must also be present, thus:
RENAME(FILENAME(RET))
If FGN and RET are both omitted, the related brackets should also be omitted, thus:
RENAME(FILENAME)
This parameter should not be used unless an OUT parameter is also present.
Note: If the final name of the output tape is of the form 'PROGRAM NAME', then the search program will be written to the tape under the name #NAME, and programs may be found from the tape by console messages of the form FI #PROG #NAME. The name thus given to the search program should not be the same as the name of any object program which is to be found from the tape. If the first eight characters of the file name are not 'PROGRAMy', then the search program will be omitted, and Executive will not be able to Find programs from the tape.
The WORK parameter: (for #XPLL and #XPLY only)

This parameter is optional. The WORK parameter defines the exchangeable disc file which is to be opened for use as the workfile, ED1. It may be included among the parameters when it is desired to open a permanent file, or a scratch file on a specific cartridge, as the work file; in its absence a scratch file on any available cartridge will be opened.

To open a permanent file as the work file, the parameter has the following format:

WORK(FILENAME(FGN1=FGN2),CSN)

where FILENAME = the existing 12-character file name. If this is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN1 = the file generation number of the existing file, in decimal, in the range 0 to 4095. It may be omitted or expressed as –1 in which case the highest numbered generation of the specified file on line will be opened.

FGN2 = the new file generation number to be written to the output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding "+" sign should also be omitted. If it is omitted the file generation number will be left unchanged.

CSN = the serial number, in octal, in the range octal 0 to octal 777777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.

To open a scratch file on a specific cartridge as the work file the parameter has the following format:

WORK(###,CSN)

where ### is invariable, and CSN is as described above. A WORK parameter with the CSN field omitted would be accepted, but the effect would be the same as if the parameter was omitted altogether.

If it becomes necessary to extend the work file, any extensions will be constrained to the cartridge which contains the last bucket of the file prior to the extension.

Up to three WORK parameters may be introduced, subsequent appearances are ignored with the accompanying message produced at the line printer:

TOO MANY WORK PARAMETERS

The first WORK parameter presented will refer to the semi-compiled file.

The second WORK parameter presented will refer to the compiler's overlay file.

The third WORK parameter presented will refer to the intermediate source file (#XPLL only) or to the compiler's temporary amendment file (#XPLY only).

For #XPLY only, note that if #XPLY is loaded directly from disc, the second WORK parameter will refer to the temporary amendment file; subsequent appearances being ignored.

The SUB parameter:

SUB(SUBFILENAME,STEERING)

where SUBFILENAME = the 12 character name of a subfile on the input tape whose contents are to constitute the input for this compilation run. The contents of all other subfiles on the input tape will be ignored. If SUBFILENAME is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

STEERING = LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST and/or BINARY and/or MAP(OFF) and/or FULLLIST(OFF) separated by commas. STEERING specifies compiler steering information which will be applied to every source segment in the subfile, in place of the information contained in each segment's steering record. It may be omitted, in which case the preceding colon should be omitted. If it is omitted, each source segment in the subfile will be treated by the compiler in accordance with the information contained in the segment's steering record.
If STEERING specifies OBJECT (or a facility which implies OBJECT) and the subfile includes semi-compiled segments, then all such segments will be included in the OBJECT program, irrespective of whether or not #SEMISTEER had been present when the input file was written (see Chapter 10).

Not more than one SUB parameter may be present in a run. If no SUB parameter is presented for a run of a magnetic tape input compiler, then all the subfiles on the input tape are examined and treated in accordance with the steering information they contain.

6 The LIB parameter: 

   LIB(FIILENAMEN FGNN/RSNN) TSN

where FILENAME = one name of the library tape.

   FGN = the file generation number of the library tape. If this is omitted or expressed as zero
   the file generation number of the tape sequence is not checked.

   RSN = the reel number of the library tape. This may be omitted (together with the preceding solidus) or expressed as zero.

   TSN = the tape serial number of the library tape. This may be omitted (together with the preceding comma) or expressed as zero.

If FGN is omitted and RSN is present, the solidus must also be present, thus:

   LIB(FIILENAME(RSN) TSN)

If FGN and RSN are both omitted, the related brackets should also be omitted, thus:

   LIB(FIILENAME TSN)

The library tape will be opened to conform with the magnetic tape opening standards.

If the compiler is loaded from magnetic tape and this parameter appears, the following message will be output on the line printer:

   INVALID PARAMETER COMBINATION

together with an associated message on the console typewriter:

   HALTED:- PE

7 The AMEND parameter: 

   (for #XPLY only)

This facility can be used to make small scale amendments to a PLAN source program held on magnetic tape, without doing a COSY update run.

The source tape itself will not be altered but the compilation will be changed by amendment parameters read from cards or paper tape.

The system will be tied to the compilation of one selected subfile, specified by an AMEND instead of a SUB parameter.

   AMEND (SUBFILE NAME: LIST, OBJECT, ETC)

where SUBFILENAME = the 12 character subfilename to be amended.

   LIST,OBJECT,etc. = the steering line parameters required during the compilation run. The presence of these parameters will override any other steering line parameters given for the named subfile. Note, however, that the presence of the AMEND parameter will automatically provide a steering line OBJECT for the subfile, and list the segments that are to be amended, whether or not they are given in the AMEND parameter.

If LIST, OBJECT, etc. are omitted, the colon must also be omitted, thus:

   AMEND(SUBFILENAME)

Parameters defining the amendments should be placed after the terminator PEND. The amendments must be specified in segment order within the subfile and in line order within the segment. Amendments to a segment should be introduced by the line:

   label operand

   #IDENTITY /SEGMENTNAME
and the #ALTER directive should follow this line. Any lines following a #ALTER directive cause the named segment to be amended until a #ALTER, #IDENTITY or #FINISH occurs. #IDENTITY indicates the end of amendments to the previously named segment. #FINISH indicates the end of all amendments.

The entire amendment file whilst being read in on cards/paper tape will be listed on the line printer and written to an E.D.S./Twin-E.D.S. file.

Any incorrect parameter lines, or any insertion line for which the appropriate preceding parameter was omitted, will be flagged by an asterisk in the left-hand margin; so too will be the insertion of a #STOP or any amendment following an inserted #END for any particular segment.

During compilation, lines that are being inserted by spot amendments will be listed with ←→ in the line number fields; whilst lines which are being omitted from the compilation will be listed with the line numbers in the usual place (print positions 1 to 4). Print positions 5 to 8 will be γ***γ

There will be a two line feed on the line printer before the omitted line(s) and a two line feed after the omitted line(s).

Any flagged amendments will cause the compiler to halt and the following message will be output on the console typewriter:

HALTED-PE

(together with an associated message on the line printer:

ERROR IN ABOVE AMENDMENTS

The maximum number of segments which can be amended is 41. Attempts to amend any more will cause the compiler to halt as above and output the following message on the line printer:

AMENDMENT FOR > 41 SEGMENTS

There is no recovery for this situation.

In the event of the segment name being erroneously specified (in the Operand field of a #IDENTITY parameter), or a #END terminator presented as an amendment followed by a further amendments for a given segment, the compilation will continue but no amendments will be made to that or any subsequent segment. At the end a SERV message will be produced at the console typewriter, with an associated message on the line printer:

AMENDMENT SEGMENT NOT FOUND

or

AMMEND. ATTEMPTED AFTER #END

8 The parameter terminator:

PEND

Parameters for PLAN 3 Compilers with Cassette Tape Output

OUTPUT FILE PARAMETERS

Any PLAN 3 compiler with cassette tape output may optionally be specified with an OUT parameter specifying the tape to be used for output. In the absence of an OUT parameter, the first available tape labelled SCRATCH TAPE is opened as the output tape. If an OUT parameter is present, it may optionally be accompanied by a REN parameter specifying the new header label information to be written to the output tape. In the absence of a REN parameter the output tape is relabelled PROGRAM XPKP by #XPLS or SEMICOMPTAPE by #XLPR, with a file generation number of 0 and a retention period of 4095 days; this applies whether the OUT parameter was also absent or not.

INPUT FILE PARAMETERS

An input file parameter is not acceptable to #XPLS, as this compiler has basic peripheral input.

With #XPLR, unless entry point 22 is used, an IN parameter is required to specify the input tape. If the IN parameter is omitted, the compiler outputs the error message 'HALTED PE' (#2400 4645) on the console.

PARAMETER TERMINATOR

The parameter terminator for cassette tape compilers is PEND. It is required for #XPLR, and is optional for #XPLS if parameters are present.
PARAMETER SPECIFICATIONS

The formats of the IN, OUT, REN and PEND parameters are identical with those described for magnetic tape output compilers in the previous subsection of this chapter. The last paragraph on the OUT parameter in that subsection does not apply in detail to machines without a console typewriter: on such machines the output tape must be loaded on station 3 for #XPLS or on station 2 for #XPLR, and Executive goes straight to the requisite station. If a parameter of the form OUT (#####.TSN) is used, and after the tape on the output station has been opened and relabelled SCRATCH TAPE the check on the tape serial number fails, then the compiler unloads the tape and an Executive message is output asking for the correct tape to be loaded.

Parameters for PLAN 3 Compilers with Disc Output

The files for the compilers #XPLM and #XPLX are on exchangeable disc. With the compilers #XPLF and #XPLZ, however, files may be on either exchangeable disc or fixed disc, and the input and output files need not necessarily be on the same type of disc. Where relevant, therefore, alternative forms of the file definition parameters are provided, and the choice of medium is indicated by the form of the parameter used to specify the particular file.

The parameters available with PLAN 3 disc output compilers are as follows:

1. **INE or INF**

   This parameter defines the disc file which is to be opened for input. It is not applicable to #XPLM or #XPLF, which have basic peripheral input, but is compulsory for #XPLX and #XPLZ.

   With #XPLZ, INE is used to specify an exchangeable disc input file, or INF is used to specify a fixed disc input file. If IN is used it will be treated as INE.

   With #XPLX INE is used to specify the exchangeable disc input file; the compiler will, however, accept either IN or INF and treat it as though it were an INE parameter.

   The parameter has the following format:

   INE(FILENAME(FGN),CSN) or INF(FILENAME(FGN))

   where

   - **FILENAME** = the existing 12 character file name. If this is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

   - **FGN** = the file generation number, in decimal, in the range 0 to 4095. It may be omitted, in which case the associated brackets should also be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on line will be opened.

   - **CSN** = the serial number, in octal, in the range octal 0 to octal 777777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.

   The check on the cartridge serial number is performed by the compiler program after Executive has opened a file of the specified name and generation number. An attempt to open the highest generation of a file on a particular cartridge by specifying CSN and omitting FGN will fail, therefore, if another cartridge containing a higher generation of the file is on line at the same time.

2. **OUTE or OUTF or APPE or APPF**

   This parameter defines the disc file which is to be opened to receive the unconsolidated semi-compiled output. It is not required if all the programs in the batch being compiled are only to be listed, but is compulsory if the steering information for any program or segment contains OBJECT.

   With #XPLF and #XPLZ, OUTE or APPE is used to specify an exchangeable disc file to receive the output, or OUTF or APPF is used to specify a fixed disc file to receive the output.

   With #XPLM and #XPLX, OUTE or APPE is used to specify an exchangeable disc file to receive the output; the compiler will, however, accept the OUTF or APPF forms and treat them as though they were the OUTE or APPE forms of the parameter respectively.

   All four compilers will accept OUTF or APP and treat them as though they were OUTE or APPE respectively.

   If OUTE or OUTF is used, the contents of the file will be overwritten by the compiler's output. If APPE or APPF is used, the compiler's output will be appended to the existing contents of the file, which will be
preserved; the existing contents must be in the format produced by a PLAN 3 disc output compiler with a mark number of 27 or later.

The parameter has the following format:

\[ \text{OUTE}(\text{FILENAME}(\text{FGN1} = \text{FGN2}), \text{CSN}) \quad \text{or} \quad \text{OUTF}(\text{FILENAME}(\text{FGN1} = \text{FGN2})) \]

or

\[ \text{APPE}(\text{FILENAME}(\text{FGN1}), \text{CSN}) \quad \text{or} \quad \text{APPF}(\text{FILENAME}(\text{FGN1})) \]

where \text{FILENAME} = the existing 12 character file name of a permanent file which is to be opened to receive the compiler output. If the file name is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

\text{FGN1} = the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, or if it is expressed as \(-1\), then the highest-numbered generation of the specified file on line will be opened.

\text{FGN2} = the new file generation number to be written to the output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding "=" sign should also be omitted. If it is omitted the file generation number will be left unchanged.

\text{CSN} = the serial number, in octal, in the range octal \(0\) to octal \(777777\), of the cartridge containing the file to be opened. It may be punched with or without a preceding \# or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.

If \text{FGN1} and \text{FGN2} are both omitted the associated brackets are also omitted, thus:

\[ \text{OUTE}(\text{FILENAME}, \text{CSN}) \quad \text{or} \quad \text{OUTF}(\text{FILENAME}) \]

or

\[ \text{APPE}(\text{FILENAME}, \text{CSN}) \quad \text{or} \quad \text{APPF}(\text{FILENAME}) \]

The version number of the file will be set to zero if \text{FGN2} is present, or will be increased by one if \text{FGN2} is omitted. \text{FGN2} is not available with \text{APPE} or \text{APPF}.

The check on the cartridge serial number is performed by the compiler program after Executive has opened a file as specified by the other fields of the parameter. It follows that:

(a) An attempt to open the highest generation of a file on a particular cartridge by specifying \text{CSN} and omitting \text{FGN1} will fail if another cartridge containing a higher generation of the file is on line at the same time.

(b) If the cartridge serial number check fails, any change called for by \text{FGN2} will already have been implemented.

If it becomes necessary to extend the output file, any extensions will be constrained to the cartridge which contains the last bucket of the file.

Note: Under GEORGE 2 an \text{OUTE/OUTE/OUTF} parameter may take the following form:

\[ \text{OUTF}(\text{FILENAME}(\ast), \text{CSN}) \]

or

\[ \text{OUTE}(\text{FILENAME}(\ast), \text{CSN}) \]

or

\[ \text{OUTF}(\text{FILENAME}(\ast), \text{CSN}) \quad \text{for } \#XPLF \text{ and } \#XPLZ \text{ only} \]

The compiler will attempt to open the file nominated for the semi-compiled with a generation number of 1. If this fails it will automatically attempt to open the file nominated with a generation number of 2, and so on, up to 4.

3 REN

This parameter is optional. It causes the file specified by the \text{OUTE} or \text{OUTF} parameter to be renamed.
The parameter has the following format:

REN(FILENAME(FGN/VN))

where FILENAME = the new file name which is to overwrite the existing file name on the output file. It may be up to 12 characters long. If it is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, the output file will be given a generation number of zero. FGN must not be expressed as -1.

VN = the version number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the solidsus should also be omitted. If it is omitted, the output file will be given a version number of zero.

If FGN is omitted and VN is present, the solidsus must also be present, thus:

REN(FILENAME(VN))

If FGN and VN are both omitted the associated brackets are also omitted, thus:

REN(FILENAME)

If a REN parameter is present, the file generation number and version number to be written to the output file are determined solely by the REN parameter; if, when REN is present, an OUTE or OUTF parameter specifies FGN2, that field of the OUTE or OUTF parameter will be overridden.

A REN parameter should not be used without an accompanying OUTE or OUTF parameter. The facility is not available in conjunction with APPE or APPF.

4 WORK

This parameter is optional. With #XPLM and #XPLX the WORK parameter defines the exchangeable disc file which is to be opened for use as the work file, ED1. It may be included among the parameters when it is desired to open a permanent file, or a scratch file on a specific cartridge, as the work file; in its absence a scratch file on any available cartridge will be opened.

To open a permanent file as the work file, the parameter has the following format:

WORK(FILENAME(FGN1=FGN2),CSN)

where the significances of FILENAME, FGN1, FGN2 and CSN, and the rules governing their use, are as described for the OUTE parameter above.

To open a scratch file on a specific cartridge as the work file the parameter has the following format:

WORK(####,CSN)

where #### is invariable, and CSN is as described for the OUTE parameter above. A WORK parameter with the CSN field omitted would be accepted, but the effect would be the same as if the parameter was omitted altogether.

If it becomes necessary to extend the work file, any extensions will be constrained to the cartridge which contains the last bucket of the file.

With #XPLF and #XPLZ the WORK parameter defines the disc file which is to be used by the consolidator as a work file if automatic consolidation is performed. The compiler itself does not use a work file. Automatic consolidation is governed by the BINE or BINF parameter, which also determines the medium, exchangeable disc or fixed disc, on which the consolidator's work file is opened; see the description of that parameter.

A permanent file or a scratch file may be specified for the consolidator's work file, the WORK parameter having the same formats as those described for #XPLM and #XPLX above, except that the FGN2 field is not implemented; if that field is present it will be accepted but ignored. The CSN field should not be present if the BINF parameter is in use.

If a WORK parameter is present for #XPLF or #XPLZ but automatic consolidation is not correctly specified, the WORK parameter is printed but otherwise ignored.
This parameter is not applicable to #XPLF and #XPLM, and is optional with #XPLX and #XPLZ. If present, it causes the compiler to operate as a single program compiler instead of as a batch compiler, and specifies which subfile of the input file is to be examined. This parameter may additionally include steering information which, if present, will override the steering information contained in the specified subfile of the input file. This facility is known as 'external steering'.

The parameter has the following format:

\[ \text{SUB(SUBFILENAME:STEERING)} \]

where \text{SUBFILENAME} = the 12 character name of a subfile on the input file whose contents are to constitute the input for this compilation run. The contents of all other subfiles in the input file will be ignored. If SUBFILENAME is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

\text{STEERING} = \text{LIST or SHORTLIST and/or OBJECT and/or MONITOR} separated by commas. STEERING specifies compiler steering information which will be applied to every source segment in the subfile, in place of the information contained in each segment's steering record. It may be omitted, in which case the preceding colon should be omitted. If it is omitted, each source segment in the subfile will be treated by the compiler in accordance with the information contained in the segment's steering record.

If STEERING specifies OBJECT and the subfile includes semi-compiled segments, then all such segments will be included in the OBJECT program, irrespective of whether or not #SEMISTEER had been present when the input file was written (see Chapter 11).

If the external steering is omitted and no source segment in the specified subfile has valid steering information present in its steering record, then a message is output on the line printer and SHORTLIST, OBJECT is assumed for all source segments in the subfile. A SUB parameter with invalid steering information is treated as though the external steering were omitted. 'Invalid steering' in the above contexts means that neither LIST nor SHORTLIST nor OBJECT is present.

If steering information (external or in the file, whichever may be applicable) contains unrecognisable items, then the unrecognisable items are ignored and the recognisable items are accepted.

Not more than one SUB parameter may be present in a run of #XPLX or #XPLZ. With these compilers, a SUB parameter must be present if automatic consolidation is required.

\section*{NEXT}

This parameter is optional. If present, it causes the compiler to delete itself and call in a successor program at the end of the run. If the NEXT parameter is absent, the compiler halts at the end of the run.

If the NEXT parameter is present, then at the end of the run the compiler deletes itself with a FIND message, to initiate the loading of a successor program. The compiler program provides the 'FI' portion of this message; the variable portion is obtained from the NEXT parameter. The NEXT parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests or charge numbers may be included in the parameter, in addition to the fields discussed below.

The generalized format of the NEXT parameter is:

\[ \text{NEXT(Variable portion of DELTY 'FIND' message)} \]

This may usually be formalized as:

\[ \text{NEXT(#ABCD \ #NAME N)} \]

where #ABCD = the name of the program to be brought into core when the compiler deletes itself. This will usually be one of the consolidator programs, #XPCK or #XPCL.

#NAME = the last four characters of the name of the file containing the successor program. The name of this file must be of the form PROGRAM NAME. #NAME may be omitted where NAME and ABCD are the same.
N = the amount of core into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program's request slip.

The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

7 BIN or BINE or BINF

This parameter is optional. It provides an automatic consolidation facility. It may be used by the four compilers only when an OUT or APP parameter (or variations of the OUT or APP parameters) and a NEXT parameter are also present. In addition, #XPLX and #XPLZ require a SUB parameter to be present.

The parameter has the following format:

(a) For use with #XPLM and #XPLX:
   (i) BINE
   or (ii) BINE(Filename(FGN))

(b) For use with #XPLF and #XPLZ:
   (i) BINE or BINF
   or (ii) BINE(Filename(FGN))
   or (iii) BINF(Filename(FGN))

All four compilers will accept BIN and treat it as though it were BINE.

In all cases, the parameter may be used only if the NEXT parameter specifies a suitable consolidator, for example, #XPCL. If a correct combination of parameters is present with this parameter, but the NEXT parameter specifies a program other than a suitable consolidator, then this parameter is accepted but ignored.

When correctly used, the BIN or BINE or BINF parameter causes the consolidator program, after it is loaded, to be entered without operator intervention, and so results in automatic consolidation of the single program being compiled (#XPLX and #XPLZ) or of the first object program output in this run (#XPLM and #XPLF).

For both compiler groups (a) and (b) above, if form (i) of the parameter is used the consolidated object program will be held in core store ready to run. If form (ii) of the parameter is used, the consolidated object program will be output in binary to the specified exchangeable disc file. For compiler group (b), if form (iii) of the parameter is used, the consolidated object program will be output in binary to the specified fixed disc file.

In the above parameter formats:

FILENAME = the existing 12 character file name of a permanent file which is to be opened and overwritten by the consolidator's output. If the file name is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted, in which case the associated brackets should also be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on the cartridge or unit containing the compiler's output file will be opened.

The version number of the file will be increased by one.

With #XPLM and #XPLX, if a WORK parameter is present with a BIN parameter, the work file used by the compiler is passed over to the consolidator for use as a work file.

With #XPLF and #XPLZ, if form (i) of the parameter is used, BINE specifies that the consolidator's work file is to be opened on exchangeable disc, whereas BINF specifies that the consolidator's work file is to be opened on fixed disc. If form (ii) or (iii) of the parameter is used, then the consolidator's work file is opened on the same medium as its output file. The file to be opened as the consolidator's work file may optionally be specified by a WORK parameter as described above.
If the automatic consolidation facility is used, then, unless a LIBE or LIBF parameter is present, any library subroutines required for the program must be in a disc file called SUBGROUPS-RS, in a subfile named SUBROUTINES. This disc file must be on the same medium, exchangeable disc or fixed disc, as the compiler's output file (that is, as the file specified in the OUTE or OUTF or APPE or APPF parameter).

8 LIBE or LIBF

This parameter is optional. It is applicable only when the automatic consolidation facility is specified. It instructs the consolidator to search library files other than or additional to SUBGROUPS-RS for any library subroutines required for the program.

With #XPLF and #XPLZ, LIBE is used to specify a library file on exchangeable disc, or LIBF is used to specify a library file on fixed disc.

With #XPLM and #XPLX, LIBE is used to specify a library file on exchangeable disc; the compiler will, however, accept LIBF and treat it as though it were LIBE.

All four compilers will accept LIB and treat it as though it were LIBE.

The parameter has the following format:

\[ \text{LIBE(FILENAME(FGN),SUBFILENAME) or LIBF(FILENAME(FGN),SUBFILENAME)} \]

where FILENAME = the 12 character file name of a library file which is to be opened by the consolidator. If the file name is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 4095. It may be omitted, in which case the associated brackets should also be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on line will be opened.

SUBFILENAME = the 12 character name of a subfile in the specified library file which is to be searched by the consolidator. If the subfile name is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

A maximum of six LIBE or LIBF parameters may be accepted. With #XPLF and #XPLZ, files on exchangeable disc and files on fixed disc may both be specified, up to the maximum of six in total. The libraries will be searched in the order in which the specifying parameters are read. The file name does not have to be unique in each LIBE or LIBF parameter.

A LIB parameter without the accompanying BINE/F,NEXT,OUTE/F,APPE/F and SUB (if appropriate) parameters is invalid.

If SUBGROUPS-RS is to be searched as well as other libraries, then a LIBE or LIBF parameter specifying SUBGROUPS-RS must be included.

With #XPLF and #XPLZ, if SUBGROUPS-RS is specified in a LIBE or LIBF parameter, then it need not be on the same medium as the compiler's output file.

If the program being compiled includes a COMPLETE directive, then that directive will override any LIBE or LIBF parameters, and no libraries will be searched.

If automatic consolidation is being performed and no LIB parameters are provided then the consolidator will search a subfile called SUBROUTINES which should be located in a file called SUBGROUPS-RS if software packages are required.

9 The parameter terminator.

The parameter terminator may optionally be present with the compilers #XPLF and #XPLM, which have basic peripheral input. With the compilers #XPLX and #XPLZ, which have disc input, the parameter terminator is required in order to release the reader and proceed to the next stage of the run.

The format of the parameter terminator is:

PEND
### Summary of parameters applicable to PLAN 3 compilers

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Applicable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Mandatory</strong></td>
</tr>
<tr>
<td>#XPLE</td>
<td></td>
</tr>
<tr>
<td>#XPLF</td>
<td></td>
</tr>
<tr>
<td>#XPLG</td>
<td></td>
</tr>
<tr>
<td>#XPLH</td>
<td></td>
</tr>
<tr>
<td>#XPLL</td>
<td></td>
</tr>
<tr>
<td>#XPLM</td>
<td></td>
</tr>
<tr>
<td>#XPLR</td>
<td></td>
</tr>
<tr>
<td>#XPLS</td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td>Applicable parameters</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>#XPLV</td>
<td><strong>Mandatory</strong>&lt;br&gt;IN&lt;br&gt;OUT&lt;br&gt;REN&lt;br&gt;SUB&lt;br&gt;PEND</td>
</tr>
<tr>
<td>#XPLW</td>
<td><strong>Optional</strong>&lt;br&gt;IN&lt;br&gt;OUT&lt;br&gt;REN&lt;br&gt;SUB&lt;br&gt;PEND</td>
</tr>
<tr>
<td>#XPLX</td>
<td><strong>Mandatory</strong>&lt;br&gt;IN or INE&lt;br&gt;PEND</td>
</tr>
<tr>
<td>#XPLY</td>
<td><strong>Mandatory</strong>&lt;br&gt;IN&lt;br&gt;OUT&lt;br&gt;REN&lt;br&gt;WORK&lt;br&gt;SUB&lt;br&gt;AMEND&lt;br&gt;PEND</td>
</tr>
<tr>
<td>#XPLZ</td>
<td><strong>Mandatory</strong>&lt;br&gt;IN, INE or INF&lt;br&gt;PEND</td>
</tr>
</tbody>
</table>

**STEERING LINES**

The PLAN 3 and PLAN 2 compilers offer a number of optional facilities, and it is necessary to indicate to the compiler which of these facilities are required on any particular run. This is done by means of steering lines. Steering lines are not used with PLAN 1.

PLAN 3 compilers with magnetic tape, cassette tape, exchangeable disc or fixed disc input have their steering information defined at the source file editor level (for source files on magnetic tape and cassette tape see Chapter 10, for source files on exchangeable disc and fixed disc see Chapter 11). Although defined via the source file editor, the format and effect of the parameters used are the same as those of the steering lines described below for PLAN 3 compilers with basic peripheral input. With #XPLV and #XPLW only, steering information for
a single program compilation run may alternatively be provided via the compiler parameter, as described on page 22.

The PLAN 2 compiler is described later in this chapter, but as the facilities it offers are offered also by the PLAN 3 compilers, its steering may conveniently be dealt with here. Only the first three of the steering line facilities described below are applicable to the PLAN 2 compiler, but the fourth, MONITOR, may be otherwise obtained in the manner indicated.

This section describes, firstly, the various facilities which can be requested by means of a steering line; secondly, the method of steering PLAN 3 compilers with basic peripheral input; and thirdly, the method of steering the PLAN 2 compiler.

Steering Line Facilities

Any suitable combination of the following facilities may be requested by the steering line. Not all facilities are available with all compilers; an indication is given in each case of the compilers with which the particular item is available.

LIST

If the steering line includes LIST, the compiler outputs on the line printer a listing of the whole of the source input together with an interpretation of its semi-compiled form, an indication of source errors and a summary of each segment's storage requirements combined with an analysis of all symbolic names used. The format of this listing is fully described in Chapter 9, pages 2 et seq, and an illustration appears in Appendix 5.

With #XPLE only, if switch 20 is set on, then the line printer output is suppressed, and the listing is produced on the paper tape punch.

The LIST facility is available with all PLAN 3 and PLAN 2 compilers.

SHORTLIST

If the steering line includes SHORTLIST, the compiler outputs an abbreviated listing on the line printer, as described on page 6 of Chapter 9.

LIST and SHORTLIST are alternatives and should not be present together on a steering line. If both are present, LIST is ignored and SHORTLIST is accepted.

With #XPLE only, if switch 20 is set on, then the line printer output is suppressed, and the listing is produced on the paper tape punch.

The SHORTLIST facility is available with all PLAN 3 and PLAN 2 compilers.

OBJECT

If the steering line contains OBJECT, compilers which do not have integral consolidators produce semi-compiled segments with segment leaders. These segments must be consolidated by a separate consolidator program before they can be loaded.

Compilers with integral consolidators, if OBJECT is present on the steering line, produce consolidated semi-compiled programs ready for loading. The individual segments are compiled to semi-compiled form until the #FINISH directive is read, (for compilers with magnetic tape or cassette tape input, an end-of-subfile sentinel implies #FINISH), and the program library is then searched for any required subroutines. If the program is then complete, the request slip, G.P.L. and consolidated leader are output. If the program is not complete, a warning message is displayed on the console typewriter; the request slip, G.P.L. and consolidated leader are still produced. Any object run using a program with unsatisfied cues will fail if conditions arise causing control to be passed to a missing segment.

The OBJECT facility is available with all PLAN 3 and PLAN 2 compilers, of which the following do not have integral consolidators:

#XPLE, #XPLF, #XPLM, #XPLR, #XPLX, #XPLZ, #XPLD

and the following do have integral consolidators:

#XPLG, #XPLH, #XPLL, #XPLS, #XPLV, #XPLW, #XPLY
MONITOR

If the steering line for a PLAN 3 compiler includes MONITOR, monitor points anywhere in a segment will result in the generation of the calling sequence to the MONITOR package, which will be automatically incorporated into the program on consolidation. If MONITOR is not on the steering line, any monitor points will be ignored.

The MONITOR facility is fully described in Chapter 9, page 17, and the programming of monitor points is described in Chapter 6, page 24. The MONITOR steering line is available only in PLAN 3, but in PLAN 2 (and PLAN 1) the MONITOR package can be called as an ordinary subroutine.

The MONITOR facility is available with all PLAN 3 compilers.

CHARGE=account code

This may be used as a steering line item with PLAN 3 compilers which have basic peripheral input and magnetic tape or cassette tape output. PLAN 3 compilers having magnetic tape or cassette tape input offer a similar facility, but it is specified differently (see the description of the #IDENTITY parameter on pages 7 and 24 of Chapter 10). With the stipulated compilers, if the steering line contains CHARGE=xxxxxxxx, where xxxxxxxx represents an account code containing up to eight alphanumeric characters, this code will be inserted in the request slip of the program, providing OBJECT is specified on the steering line. The account code is also given in a console type-writer message (see page 40) at the end of the compilation.

Note: The account code inserted in the request slip will be the characters immediately following the equals sign, up to the expiry of a count of eight characters or a comma, whichever is reached first. If, therefore, spaces are left between the equals sign and the first character of the account code, these spaces will be included in the character count.

The CHARGE steering line facility is available with the compilers #XPLE, #XPLF, #XPLG, #XPLH, #XPLL, #XPLM, #XPLS and #XPLY.

MAP

MAP may be used as a steering line item with PLAN 3 magnetic tape or disc compilers.

MAP implies OBJECT. With the stipulated compilers, if the steering line contains MAP, the program is compiled and consolidated as described for OBJECT above. When consolidation has been completed, a segment-by-segment analysis of the storage requirements of the consolidated program is output on the line printer. The analysis includes the octal and decimal representation of the address, relative to the program datum, allocated to each symbolic name used.

If the steering line contains LIST, MAP then the LIST compiler listing is produced, except that the analysis following the #END directive for each segment (see pages 4 and 5 of Chapter 9) is limited to the literals table and the error line; the MAP analysis of the consolidated program follows after the consolidated program summary.

Although MAP implies OBJECT, it is acceptable in combination with OBJECT.

The MAP facility is available with the compilers which produce a MAP analysis of the consolidated program:

#XPLG, #XPLH, #XPLL, #XPLV, #XPLW and #XPLY

and the following do not but will set up MAP records suitable for input to #XPSS:

#XPLF, #XPLM, #XPLX and #XPLZ.

FULLLIST

FULLLIST may be used as a steering line item with PLAN 3 magnetic tape or disc compilers.

FULLLIST implies OBJECT and MAP, but is acceptable in combination with either or both. With the stipulated compilers, if the steering line contains FULLLIST, the program is compiled and consolidated as described for OBJECT above. When the consolidation has been completed, the compiler outputs on the line printer a listing of the whole source input together with the resulting object program. The run-time address, relative to the program datum, is shown for each instruction and each item of preset data. Where the symbolic name of a data location is referenced in presets or literals, the decimal address of the referenced location is printed to the right of the octal representation of the preset or literal. Any source errors are indicated as for LIST. Heading the listing is a note of the program name, priority, total core requirement, and peripheral requirements as declared in #PERIPHERAL.
directives. Following each segment is an analysis of that segment’s storage requirements and symbolic names as produced by MAP, with the addition of a mapping of the literal table.

An illustration of a full list appears in Appendix 5.

If the steering line contains LIST, FULLLIST, the LIST compiler listing is produced (without any suppression of the segment summary) followed by the full list.

The FULLLIST facility is available with the compilers which produce a full list of the consolidated program:

#XPLG, #XPLH, #XPLL, #XPLV, #XPLW and #XPLY

and the following do not but will set up FULLLIST records suitable for input to #XPSS:

#XPLF, #XPLM, #XPLX and #XPLZ.

MAP(OFF)

MAP(OFF) may be used as a steering line item with PLAN 3 magnetic tape compilers, but will have the desired effect only if it is on the first program level steering line (compilers with basic peripheral input) or first non-blank steering record (compilers with magnetic tape input) encountered for a program.

MAP(OFF) implies OBJECT. With the stipulated compilers, MAP(OFF) on a program’s first steering line has a similar effect to MAP, except that the printing of the store map on the line printer is inhibited; instead, the store map information is retained on the compiler’s output tape, whence it may subsequently be printed by the program #XPST.

Although MAP(OFF) implies OBJECT, it is acceptable in combination with OBJECT.

The MAP(OFF) steering line facility is available with the compilers #XPLG, #XPLH, #XPLL, #XPLV, #XPLW and #XPLY.

FULLLIST(OFF)

FULLLIST(OFF) may be used as a steering line item with PLAN 3 magnetic tape compilers, but will have the desired effect only if it is on the first program level steering line (compilers with basic peripheral input) or first non-blank steering record (compilers with magnetic tape input) encountered for a program.

FULLLIST(OFF) implies MAP(OFF) and OBJECT. With the stipulated compilers, FULLLIST(OFF) on a program’s first steering line has a similar effect to FULLLIST, except that the printing of the full listing on the line printer is inhibited; instead, the full listing information is retained on the compiler’s output tape, whence it may subsequently be printed by the program #XPST.

Although FULLLIST(OFF) implies MAP(OFF) and OBJECT, it is acceptable in combination with either or both MAP, FULLLIST(OFF) is also acceptable.

The FULLLIST(OFF) steering line facility is available with the compilers #XPLG, #XPLH, #XPLL, #XPLV, #XPLW and #XPLY.

BINARY

The BINARY facility is available with PLAN 3 compilers which have magnetic tape output, and with #XPLS.

BINARY implies OBJECT. If the steering line contains BINARY, the program is compiled and consolidated as described for OBJECT above, except that the compiler will provide the overlay loader, #XPO6 (magnetic tape) or #XPP6 (cassette tape), for a non-overlay program. When the program is subsequently found, the loader, in addition to loading the program, will open a scratch tape, relabel it PROGRAM NAME (where NAME is the name of the program being loaded) and write the program to it in binary, in a form suitable for chaining.

Although BINARY implies OBJECT, it is acceptable in combination with OBJECT.

The BINARY facility is available with the compilers #XPLG, #XPLH, #XPLL, #XPLS, #XPLV, #XPLW and #XPLY.

Steering Lines for PLAN 3 Compilers with Basic Peripheral Input

#XPLF, #XPLG, #XPLH, #XPLL, #XPLM, #XPLY

PLAN 3 compilers with basic peripheral input and output on magnetic tape or output on disc may be steered both
at program level and at segment level. Each program to be compiled must be preceded by an overall (program level) steering line punched into one line of paper tape or one card, starting from character position one of the paper tape line or column one of the card. The format of this steering line is as follows:

#STEER Steering information

where Steering information = any suitable combination, relevant to the compiler in use, of the following items: LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or CHARGE=xnnnxxxxx (where xnnnxxxxx represents up to eight alphanumeric characters) and/or MAP and/or FULLLIST (and the following further steering items applicable only to #XPLG, #XPLH, #XPLL and #XPLY) and/or MAP(OFF) and/or FULLLIST(OFF) and/or BINARY. The items selected may be in any order, and must be separated from each other by commas and optional spaces. The chosen selection must start from character position 16 of the paper tape line or column 16 of the card. If paper tape is used, horizontal tabulation characters will be expanded to the format of PLAN source cards.

The overall steering line determines which compiler facilities will be called for all following segments of the program, unless overridden for particular segments by a segment level steering line, or until superseded by a later #STEER line with different steering information. Further #STEER lines may be used between later segments of a program to call different selections of facilities for later groups of segments.

Any segment of the program may be preceded by a segment level steering line, punched in the same manner as an overall steering line. The format of the segment level steering line is as follows:

#SEGSTEER Steering information

where Steering information = any suitable combination of items, relevant to the compiler in use, as described above for #STEER.

Any of the facilities described in the previous sub-section may be called at the segment level. The segment level steering line is operative only for the segment that it immediately precedes and for that segment replaces the overall steering information.

The following points should be noted:

1. CHARGE=account code may appear in any steering line at program level or segment level, but only the first CHARGE code encountered will be implemented in the request slip. Any subsequent appearance of this item will be accepted but ignored.

2. MAP and FULLLIST, whether used at the program level or at the segment level, produce printouts which relate to the current consolidation of the program only. Whereas a list or a short list of any particular segment remains valid when other segments are amended and recompiled, the validity of the store maps or full lists of all segments is liable to be affected when any segment is amended and recompiled. Similarly if a program includes any library subroutines, and is recompiled to include later versions of these subroutines, any previously existing store maps or full lists are liable to be invalidated.

If MAP or FULLLIST is used at the segment level, the resulting printout will include an indication of all segments which were submitted at this consolidation but for which MAP or FULLLIST were not requested.

The above remarks apply equally to MAP(OFF) and FULLLIST(OFF).

If #SEGSTEER is used with a blank operand it is flagged as an O error in the compiler listing and is otherwise ignored.

Any segment of the program may have either of either a #STEER line and a #SEGSTEER line preceding it, except that the first segment must have a #STEER line and may have a #SEGSTEER line. If both lines are present before any segment (including the first) they may be in either order. If more than one steering line of the same type, program level or segment level, is read preceding a particular segment, the first of that type is accepted and the subsequent ones are listed and error flagged on the line printer, but not implemented. If the first segment of a program is not preceded by a program level steering line, a message is output on the line printer, and SHORTLIST, OBJECT is assumed.

If the operand of a steering line contains any item which is not recognisable as steering information, (including any item which, although a valid steering item in PLAN 3, has not been implemented on the particular compiler in use), the recognisable items are accepted and no action is taken on the unrecognisable items.

If PLAN 3 compiler parameters, as described in the previous section of this chapter, are employed, those parameters must precede the first steering line input.
#XPLE, #XPLS

PLAN 3 compilers with basic peripheral input and output on basic peripherals or output on cassette tape may be steered at the program level only. Each program to be compiled must be preceded by a steering line punched into one line of paper tape or one card, starting from character position one of the paper tape line or column one of the card. The format of the steering line is as follows:

\[ \text{#STEER Steering information} \]

where \( \text{Steering information} = \text{LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or CHARGE=xxxxxxxx} \) (where \( xxxxxxxx \) represents up to eight alphanumeric characters) and/or (for #XPLS only) \( \text{BINARY} \). The items selected may be in any order, and must be separated from each other by commas and optional spaces. The chosen selection must start from character position 16 of the paper tape line or column 16 of the card. If paper tape is used, horizontal tabulation characters will be expanded to the format of PLAN source cards.

The steering line determines which compiler facilities will be called for all segments of the program. It is not permissible to use further steering lines between segments of a program.

Error indication is given on the console if the first segment of a program is not preceded by a program level steering line.

If the operand of a steering line contains any item not recognisable as applicable to these compilers, the recognisable items are accepted and no action is taken on the unrecognisable items.

With #XPLS, if compiler parameters as described in the previous section of this chapter are employed, they must precede the steering line.

**Steering Lines for the PLAN 2 Compiler #XPLD**

The PLAN 2 compiler #XPLD will accept only an overall (program level) steering line. This steering line must be the first item input, that is, immediately preceding the first #PROGRAM line. The steering line must be punched into one paper tape line or one card, starting at character position one of the paper tape line or column one of the card. It consists of any of the following items:

- \( \text{LIST or SHORTLIST and/or OBJECT} \)
- or a combination of them separated by a comma and optional spaces. The items may be in any order.

**PLAN 3 and PLAN 2 Compiler Steering Line Facilities**

<table>
<thead>
<tr>
<th>COMPILER</th>
<th>LIST</th>
<th>SHORTLIST</th>
<th>OBJECT</th>
<th>MONITOR</th>
<th>CHARGE</th>
<th>MAP</th>
<th>FULLIST</th>
<th>MAROFF</th>
<th>FULLISTOFF</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XPLF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>•</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLV</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLW</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>•</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLZ</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPLD</td>
<td>✓</td>
<td>✓</td>
<td>•</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

✓ = Available as a steering line facility.

* = Not available as a steering line facility but available by other means.

- = Not available.
OPERATING INSTRUCTIONS FOR PLAN 3 COMPILERS

Operating Instructions for the PLAN 3 Compiler #XPLE

1. Load the compiler into core store by an appropriate LOAD or FIND message.
2. Load the source program, preceded by its steering line, on the paper tape reader or card reader.
3. If any switches are to be set on, set them by typing in the console message:

   ON #XPLE n

   where n is as many as are appropriate of the following switch numbers:

   18  if LIST or SHORTLIST is on the steering line, and a throw to a new page is required when a punching in channel 8 of the control loop is detected (instead of when a count of 60 lines expires).
   20  if LIST or SHORTLIST is on the steering line, and the listing is to be output to a paper tape punch instead of to a line printer.
   21  if LIST or SHORTLIST is on the steering line, and a 96 print position printer is to be used.
   22  if OBJECT is on the steering line, and the semi-compiled object program output is to be on cards instead of on paper tape.

4. Activate the compiler by typing in one of the following console messages:

   (a) If the basic peripheral input is initially on paper tape:

      GO#XPLE 20

   (b) If the basic peripheral input is initially on cards:

      GO#XPLE 21

   The basic peripheral input medium may be changed at any time by a #SWITCH directive.

5. At the end of the compilation of the program, the compiler outputs the following console messages:

   (a) If the compilation was successfully completed and no errors were detected in the source program:

      0#XPLE;DISPLAY:--END OF #NAME
      0#XPLE;DISPLAY:--COMPiled #NAMExxxxxxxxxxxx

   (b) If the compilation was successfully completed but errors were detected in the source program:

      0#XPLE;DISPLAY:--END OF #NAME - ERRORS
      0#XPLE;DISPLAY:--COMPiled #NAMExxxxxxxxxxxx

In the above messages:

   AAAAAAAAA = the total of the number of lines printed and the number of lines punched during the compilation of the program.

   #NAME = the name of the program compiled.

   XXXXXXXX = if present, an 8-digit account code (see under the CHARGE steering line facility, page 36).

6. At the end of the run the basic peripherals are released and the following message is output on the console typewriter:

   0#XPLE;HALTED:--END OF COMPILATION

7. Either delete #HALTED or return to step 2 to compile another program.

Operating Instructions for the PLAN 3 Compilers Other than #XPLE

(Lights displays and handswitch messages for cassette tape compilers when run in machines without a console typewriter are shown in parentheses.)

1. Set up any magnetic tapes, cassette tapes or exchangeable discs required for the particular run.
2. Load the compiler into core store by an appropriate FIND or LOAD message.
3 Load the parameters and/or steering lines and source program, as applicable to the particular compiler, on the paper tape reader or card reader.

4 If a 96 print position line printer is to be used, input the console message:

   ON #XPL? 21  (#0400 0025)

5 If it is required that any compiler lists or store maps which may be requested by steering information should throw to a new page when a punching in channel 8 of the control loop is detected (instead of when a count of 60 lines expires), input the console message:

   ON #XPL? 18  (#0400 0022)

6 Activate the compiler by typing in one of the following console messages:

   (a) If the basic peripheral input is initially on paper tape:

       GO #XPL? 20  (#0100 0024)

   (b) If the basic peripheral input is initially on cards:

       GO #XPL? 21  (#0100 0025)

   (c) If the input file is a magnetic tape or cassette tape called COMPILE FILE, and there is no basic peripheral input:

       GO #XPL? 22  (#0100 0026)

       This input message can apply only to the compilers

       #XPLV, #XPLW, #XPLY and #XPLR

   The basic peripheral input medium may be changed at any time, by a #SWITCH directive.

7 At the end of the compilation of each program, the compiler outputs one of the following console messages:

   (a) Compilers other than cassette tape compilers:

      (i) If the compilation was successfully completed and no errors were detected in the source program:

          0#XPL?:DISPLAY::--COMP OK AAAAAAAA#NAMEXXXXXXXX

      (ii) If the compilation was successfully completed but errors were detected in the source program:

          0#XPL?:DISPLAY::--COMP ER AAAAAAAA#NAMEXXXXXXXX

      (iii) If the compilation was abandoned:

          0#XPL?:DISPLAY::--SERV ER AAAAAAAA#NAMEXXXXXXXX

       A message giving the reason for the abandonment is output on the line printer.

In the above messages:

   AAAAAAAA = the total of the number of lines printed and the number of blocks written to magnetic media during the compilation of the program.

   #NAME = the name of the program compiled.

   XXXXXXXXX = if present, an 8-digit account code (see under the CHARGE steering line facility, page 36).

   (b) Cassette tape compilers (#XPLR, #XPLS):

      (i) If no errors were detected in the source program:

          DISPLAY EP  (#2400 4560)

      (ii) If errors were detected in the source program:

          DISPLAY EE  (#2400 4545)

8 At the end of the run the basic peripherals are released and one of the following messages is output on the console:

   (a) Compilers other than cassette tape compilers:
(i) If a successor program has been specified by a compiler parameter:

0#XPL?; DELETED:-Fl . . . .

(The variable part of the DELTY 'FIND' message is inserted as specified in the compiler parameter.)

(ii) If no successor program has been specified:

0#XPL?; HALTED:- END OF BATCH

for magnetic tape compilers, or:

0#XPL?; HALTED:- END OF RUN

for disc compilers.

(b) Cassette tape compilers (#XPLR, #XPLS):

HALTED EB

(#2400 4542)

9 Except where a successor program has been specified by compiler parameters, either delete #XPL?, or repeat steps 1, 3 and 6 to compile further programs. For #XPLX and #XPLZ step 2 will also have to be repeated.

EXCEPTION CONDITIONS, PLAN 3 COMPILERS

This section describes the error messages which may be output on the console typewriter and the diagnostic information which may be output on the line printer by the PLAN 3 compilers. It also describes the compiler post-mortem procedures. It does not include the various Executive messages that may arise in the course of a compilation; details of these may be found in the appropriate console operating manual. In the event of an Executive message indicating a hardware fault from which there is no recovery occurring during a batch compilation run, the output tape or output file will be of use up to and including the program prior to that during the compilation of which the fault occurred.

The section is divided into subsections dealing separately with those compilers which have object output on basic peripherals, on magnetic tape, on cassette tape and on discs.

PLAN 3 Compilers with Basic Peripheral Output (#XPLE)

The following messages may be output on the console typewriter by #XPLE:

DISPLAY:- PL

A paper low condition has been detected on the line printer. The line printer is disengaged. The operator should replenish the paper before reallocating the printer to continue the run.

HALTED:- CP
HALTED:- CR
HALTED:- LP
HALTED:- P0
HALTED:- P1
HALTED:- TR

One of these messages is output if the compiler tries to allocate a peripheral which is not available. (P0 indicates a paper tape punch for semi-compiled object program output, P1 indicates a paper tape punch for listing.) Make the required peripheral available and resume by GO #XPLE.

HALTED:- INCORRECT STEERING LINE

No recognisable steering line has preceded the source program. Provide a correct steering line, reload the reader and restart by GO #XPLE 20 (input on paper tape) or GO #XPLE 21 (input on cards).

HALTED:- CONVERGING TABLES #NAME

The combined symbol and branch ahead table has been filled and compilation cannot continue. On a single program processor, type in GO #XPLE; the rest of the segment will be ignored and the compiler will proceed to the next segment. On a multiprogram processor, if more core store can be made available by deleting an idle program, delete the idle program and type in GO #XPLE 27; the compiler will allocate itself more core store and continue compilation. If more core store cannot be made available, act as for a single program processor.

The following messages may be output on the line printer by #XPLE:

READER FAILED HERE

There has been an Executive FAIL message in respect of the input device.
(the paper tape reader or card reader) and the operator has typed
  GO #XPLE to continue the run. This message is printed to indicate the
line on which the failure occurred, and compilation resumes from the
next line. The message is flagged with O and P error flags (see Chapter 9).

*TABLE EXCEEDED 1600 WORDS
AT LINE xxxx
If the compiler is suspected of looping, first output word 8 of the console by typing in:
  OU #XPLE 8
then continue with the post-mortem procedure.
The post-mortem procedure, should the compiler go ILLEGAL or be suspected of looping, is to type in:
  GO #XPLE 29
The end of the post-mortem is signalled by the console output message:
  O#XPLE; HALTED:-- PM
The post-mortem output and console log should be forwarded through the normal channels for software error.

PLAN 3 Compilers with Magnetic Tape Output (#XPLG, #XPLH, #XPLL, #XPLV, #XPLW, #XPLY)
The following messages may be output on the console typewriter by the compilers #XPLG, #XPLH, #XPLL,
#XPLV, #XPLW and #XPLY.

DISPLAY:- BC
The program being consolidated has an unsatisfied cue. If the unsatisfied
cue is a name defined in a #OVERLAY directive the program is aban-
donied, the message OVERLAY BLANK CUE xxxx (for #XPLL only)
or OVERLAY BLANK CUE:-- xxxxxxxxxxx, where x . . x is the
name of the first unsatisfied overlay cue encountered in the program, is
output on the line printer, the SERV display (see step 7 of the operating
instructions) is output for the program, and the compiler proceeds to the
next program. Otherwise, the missing name is given a value of #75777, the
COMP OK display is output for the program, and consolidation continues.

If the program contains both overlay blank cues and other unsatisfied
cues, then the DISPLAY:- BC message is output twice.

DISPLAY:- FF
The compiler’s post-mortem procedure is in operation but the post-
mortem output will not be complete. The post-mortem continues.

DISPLAY:- PL
A paper low condition has been detected on the line printer. The line
printer is disengaged. The operator should replenish the paper before
reallocating the printer to continue the run.

DISPLAY:- SB
An end-of-tape marker has been detected on one of the magnetic tapes,
so the rest of the batch has been abandoned. A SERV display (see step 7
of the operating instructions) is output for the program being compiled
at the time of abandonment. The message UN.n FULL #NAME NOT
COMP. (where n is the compiler program’s relative unit number of the
desk containing the full tape, and #NAME is the name of the program
being compiled at the time of abandonment) is output on the line printer.

HALTED:- CR
HALTED:- LP
HALTED:- TR
One of these messages is output when the compiler tries to allocate a
peripheral which is not available. Make the required peripheral available
and resume by GO #XPL?.

HALTED:- L/BLK UNIT n
A long block has been detected on a magnetic tape. The compiler pro-
gram’s relative unit number of the deck containing the faulty tape is
given by n. Abandon the run.

HALTED:- PAR/F UNIT n
There has been a parity failure on a magnetic tape, and the operator has
attempted to resume the run by GO #XPL?. The compiler program’s
relative unit number of the deck containing the failed tape is given by n.
Abandon the run.

HALTED:- OE
The compiler cannot find one of its overlays. Repeat the run with the P.L.T.
on a different deck. If the halt recurs, the library tape should be recreated.
HALTED:-- PE

There is an error in the compiler parameters. A line printer message indicates the error. Correct the parameter, reload all the parameters in the reader and restart by GO #XPL? 20 (parameters on paper tape) or GO #XPL? 21 (parameters on cards).

The following messages may be output on the console typewriter by the compilers #XPLL and #XPLY.

HALTED:-- NEEDS DISC NNNNN

A file with a filename specified in a WORK parameter was opened, but the check on the cartridge serial number failed, so the file has been closed. Make the correct cartridge available and resume by GO #XPLL or GO #XPLY.

HALTED:-- On

A named disc file or sufficient scratch space is not available on line. A message at the line printer will indicate the appropriate situation.

n = Open failure type from Executive reply (see 1900 Series manual Direct Access). Make specified file available if possible, depending upon the WORK parameters and reply type; restarting by GO #XPLL or GO #XPLY.

The following messages may be output on the line printer by the compilers #XPLL and #XPLY.

DISC TRANSFER ERROR X
UNIT N

X = Transfer failure reply (see 1900 Series manual Direct Access). The next program is examined if possible.

FILENAME XXXX N NOT OPENED
SCRATCH FILE N NOT OPENED

The compiler has not found either the specified filename N or sufficient space on the cartridge to open a scratch file N (= 1 or 2).

HALTED:-- On will appear at the console typewriter (see Console messages above for further details).

FILENAME XXXX *NNNNNN
AS X
SCRATCH FILE *NNNNNN
AS X
OVERLAYS NOT MOVED TO EDS
FILE EXTENSION ERR X
UNIT N

*NNNNNN = cartridge on which file X (= 0,1 or 2) is opened. The compiler has failed either to find enough space, or a specific file, or enough space on that file to allocate as its overlay file. Instead of transcribing its overlay it will overlay itself from the program library tape for the duration of the run.

N = compiler's specific disc file full (1 or 2).
X = Extension failure reply (see 1900 Series manual Direct Access).

Next program is examined if possible.

FILENAME XXXX EXT'D. TO NNNNN
FILENAME XXXX FINAL BKT. NNNNN

At the end of batch for all named files extended during run.

At the end of batch for all named files used.

TOO MANY WORK PARAMETERS

More than three WORK parameters have been presented or if #XPLY is loaded directly from disc, more than two WORK parameters have been presented. Superfluous parameters are ignored; compilation continues normally.

The following messages may be output on the line printer by the compiler #XPLY.

ERROR IN ABOVE AMENDMENTS
AMENDMENTS FOR > 41 SEGMENT
AMENDMENT SEGMENT NOT FOUND
AMDMNT_ATTEMPTED AFTER #END

INVALID PARAMETER COMBINATION

A format error has been detected in reading the runtime amendments.

1 LIB parameter submitted when loading is initially from magnetic tape.

2 SUB parameter submitted with a WORK parameter for the temporary amendment file.

3 An IN parameter, implying card/paper tape input has been submitted with a SUB or AMEND parameter.
NO VALID STEERING FOR S/FILE

The subfilename specified in the SUB parameter has been found but no valid steering information is present.

SUBFILE NAME NOT FOUND

The subfilename specified in the SUB/AMEND parameter cannot be found on the input file.

The following messages may be output on the line printer by the compilers \#XPLG, \#XPLH, \#XPLL, \#XPLV, \#XPLW and \#XPLY:

CNSL. TABLE FULL \#NAME

The consolidated cue list has become full and consolidation cannot continue. The program \#NAME is abandoned, the SERV message is output on the console typewriter and the compiler proceeds to the next program in the batch.

ERROR IN PARAMETER ABOVE

There is an error in the compiler parameter last printed. The compiler outputs the console message HALTED:-- PE. See under that console message above.

GROUP name NOT FOUND

The subroutine group 'name' cannot be found on the library tape. Consolidation continues, but any instructions calling subroutines from the missing group are given an operand of \#75777.

LOWER LIMIT VIOLATION \#NAME

\#NAME has allocated more than 4096 words of lower store. Consolidation continues, but whether the program can be run or not depends on how the offending lower store is addressed.

MAP/FULLLIST TABLE FULL \#NAME

The map/fulllist table for \#NAME has been filled, and the line printer output cannot be completed. The program is abandoned, the SERV message is output on the console typewriter and the compiler proceeds to the next program in the batch. Note: although the SERV message is output, the object program is in this case usable; it is only the map/full listing which is not completed.

OVERLAY BLANK CUE:--
xx

OVERLAY BLANK CUES xxx

See under the console message DISPLAY:-- BC.

RENAME BUT NO OUT PARAMETER

A REN parameter is present among the compiler parameters, but it is not accompanied by an OUT parameter. The compiler outputs the console message HALTED:-- PE. See under that console message above.

*TABLE EXCEEDED nnnn WORDS AT LINE xxxx

For an explanation of this message see page 20 of this chapter.

TABLE FULL \#NAME

The combined symbol and branch ahead table has been filled and compilation cannot continue. The program is abandoned, the SERV message is output on the console typewriter and the compiler proceeds to the next program.

TWO PARAMETERS SAME TYPE

Two or more compiler parameters of the same type have been input. The second and subsequent parameters of the repeated type are ignored.

UN.n FULL \#NAME NOT COMP.

See under the console message DISPLAY:-- SB.

The following messages may be output on the line printer by the compilers \#XPLG, \#XPLH, \#XPLL and \#XPLY:

NO PROGRAM LEVEL STEERING

A program is not preceded by a \#STEER steering line. The compiler assumes SHORTLIST, OBJECT as the program level steering information for the program, and proceeds. When the program has been compiled, the COMP ER version of the end of compilation message is displayed on the console typewriter, irrespective of whether other errors were detected in the source program.

READER FAILED HERE

There has been an Executive FAIL message in respect of the input device (the paper tape reader or card reader) and the operator has typed GO \#XPL7 to continue the run. This message is printed to indicate the line on which the failure occurred, and compilation resumes from the next line. The message is flagged with O and P error flags (see Chapter 9).
The following messages may be output on the line printer by the compilers #XPLV and #XPLW:

**NO VALID STEERING FOR S/FILE**  The subfile specified in the SUB parameter has been found, but there is no valid steering information present. The compiler proceeds to its end routines and outputs the console message HALTED:: END OF BATCH. Replace the SUB parameter by one containing steering information, and re-run.

**SUBFILE NAME NOT FOUND**  The subfile name specified in the SUB parameter cannot be found in the input file. The compiler proceeds to its end routines and outputs the console message HALTED:: END OF BATCH. Submit fresh parameters and re-run.

If a compiler in this group is suspected of looping, first output word 8 on the console by typing:

```
OU #XPL? 8
```
then continue with the post-mortem procedure.

The post-mortem procedure, should the compiler go ILLEGAL or be suspected of looping, is to input the console message:

```
GO #XPL? 29
```
In certain circumstances the compiler will automatically post-mortem itself.

The end of the post-mortem is indicated by the console message:

```
0#XPL?: HALTED:: END OF PM
```
The line printer output and the console log should be forwarded through the normal channels for software error. If it ever becomes necessary to inhibit the compiler's automatic post-mortem, this may be done by setting switch 14 on before commencing the run. Should the circumstances then arise which, were switch 14 not on, would initiate the automatic post-mortem, the compiler will halt with the console message:

```
0#XPL?: HALTED:: PM
```
In this event, an Executive core print of the compiler's area of store should be taken and forwarded, together with the console log and printouts of the magnetic tapes used, through the normal channels for software error. Typing

```
GO #XPL?
```
in these circumstances will cause the compiler's post-mortem routines to be entered.

**PLAN 3 Compilers with Cassette Tape Output (#XPLR, #XPLS)**
(Lights displays for machines without a console typewriter are shown in parentheses.)
The following messages may be output on the console by #XPLR and #XPLS:

**DISPLAY:: PL**  This message appears on the console if listing is requested and a paper low condition is detected on the line printer. The printer is disengaged. The operator should replenish the paper and re-engage the printer to continue the run.

- **HALTED CR**  (#2400 4362)  One of these messages is output if the compiler tries to allocate a peripheral which is not available. Make the required peripheral available and resume by GO.
- **HALTED LP**  (#2400 5460)  An end-of-tape marker has been detected on one of the cassette tapes, so the rest of the batch has been abandoned.
- **HALTED TR**  (#2400 6462)  The parameter just read is in error. Correct the parameter, reload all the parameters in the reader, and resume by GO.
- **HALTED E1**  (#2400 4501)  The compiler is unable to find one of its overlays. Abandon the run and investigate.
- **HALTED FE**  (#2400 4645)  There has been an error in reading from cassette tape. Abandon the run.
- **HALTED OE**  (#2400 5745)  The combined symbol and branch ahead table has been filled and compilation cannot continue. GO will cause the compiler to ignore the rest of the segment and to proceed to the next segment.
- **HALTED PE**  (#2400 6045)  The compiler is unable to find one of its overlays. Abandon the run and investigate.
- **HALTED TF**  (#2400 6446)  The compiler is unable to find one of its overlays. Abandon the run and investigate.
HALTED NEEDS TAPE nnnnn (#2400 6764) A tape with a file name specified in a parameter, or a tape with an expired retention period as called for by a parameter, was opened, but the check on the tape serial number failed; so the tape has been unloaded. Make the correct tape available and resume by GO. On a machine without a console typewriter, the HALTED NEEDS TAPE nnnn message will be printed on the line printer, and the message HALTED WT displayed on the lights.

The following further message may be output on the console by #XPLR only:

HALTED CI (#2400 4351) The format of the input cassette is incorrect and compilation cannot continue. Abandon the run.

The following further messages may be output on the console by #XPLS only:

DISPLAY BC (#2500 4243) There is an unsatisfied cue in a program. Consolidation continues, any missing name being given a value of #75777.

DISPLAY LL (#2500 5454) A program has allocated more than 4096 words of lower store. Consolidation continues, but whether the program can be run or not depends on how the offending lower store is addressed.

DISPLAY NP (#2500 5660) There is no PLAN subroutine group on the P.L.T. Calls to library subroutines are given an operand value of #75777. If other programs in the batch need library subroutines there is little point in continuing, so after the next DISPLAY EP or DISPLAY EE message the run should be terminated by the input message SU.

DISPLAY OB (#2500 5742) A segment defined under #OVERLAY is missing. Consolidation of the program is abandoned, the message DISPLAY EP or DISPLAY EE is output, and the compiler proceeds to the next program of the batch.

HALTED CF (#2400 4346) The consolidated cue list has been filled and consolidation cannot continue. GO will cause the program to be abandoned, the message DISPLAY EP or DISPLAY EE to be output, and the compiler to proceed to the next program of the batch.

HALTED IS (#2400 5163) No recognisable steering line has preceded a source program. Provide a correct steering line, reload the reader, and resume by GO 20 (input on paper tape) or GO 21 (input on cards).

Any other halt Take a post-mortem as described below.

The following message may be output on the line printer by #XPLR and #XPLS:

*TABLE EXCEEDED 800 WORDS AT LINE xxxx For an explanation of this message see page 20 of this chapter.

The following message may be output on the line printer by #XPLS:

READER FAILED HERE There has been an Executive FAIL message in respect of the input device (the paper tape reader or card reader) and the operator has typed GO #XPLS to continue the run. This message is printed to indicate the line on which the failure occurred, and compilation resumes from the next line. The message is flagged with O and P error flags (see Chapter 9).

If #XPLR or #XPLS is suspected of looping, first output word 8 on the console by the input message:

    OU 8 (#0200 0010)

then continue with the post-mortem procedure.

The post-mortem procedure, should the compiler go ILLEGAL or be suspected of looping, is to input the console messages:

    ON 10 11 12 13 (#0400 0012)
    (#0400 0013)
    (#0400 0014)
    (#0400 0015)
    GO 29 (#0100 0035)

Chapter 7 47
All cassette tapes except the P.L.T. will be printed, and the final output tape will be left scratch. The end of the post-mortem is signalled by the console output message:

HALTED PM  (#2400 6055)

If an error occurs in the post-mortem, so that it cannot be completed, the end message is:

HALTED En  (#2400 000n)

where n = a digit in the range 1 to 4.

In either case the post-mortem output and console log should be forwarded through the normal channels for software error.

PLAN 3 Compilers with Disc Output (#XPLF, #XPLM, #XPLX, #XPLZ)

The following messages may be output on the console typewriter by the compilers #XPLF, #XPLM, #XPLX, and #XPLZ. They are described as for #XPL and #XPLZ; references to INE or INF parameters and input files should be ignored for #XPLF and #XPLM.

DISPLAY:- FILENAME name *nnnnnn USED AS x
HALTED:- NEEDS DISC *nnnnnn  

This pair of messages indicates that an exchangeable disc file with the file name specified in the INE or the OUTE or APPE parameter was opened, but that the check on the cartridge serial number, *nnnnnn, failed, so the file has been closed. x = 4 for the input file and 2 for the output file.

Make the correct cartridge available, and resume by the console message GO #XPL?.

DISPLAY:- On  
The compiler has failed to rename the specified disc file and the compilation continues.

DISPLAY:- PL  
A paper low condition has been detected on the line printer. The line printer is disengaged. The operator should replenish the paper before reallocating the printer to continue the run.

HALTED:- CR  
HALTED:- LP  
HALTED:- TR  

One of these messages is output if the compiler tries to allocate a peripheral which is not available. Make the required peripheral available and resume by GO #XPL?.

HALTED:- IF  
The file specified by an INE or INF or an APPE or APPF parameter is not in the correct format. A line printer message indicates the type of format error. Abandon the run.

HALTED:- NEEDS DISC *nnnnnn
HALTED:- On  
See under DISPLAY:- FILENAME name *nnnnnn USED AS x.

A required file (permanent or scratch) has not been opened, for a reason indicated by n. n = the contents of bits 6 to 23 of the reply word printed as a decimal number.

For the significance of the reply, see the manual 'Direct Access'. A line printer message gives further information.

The cause of the failure may be:

1 the correct cartridge not having been loaded.
2 a fault in a parameter.
3 insufficient space being left on the cartridge to open a scratch file.
4 the current multi-programming situation.

As appropriate, load the correct cartridge, provide an appropriate new parameter or await a change in the multi-programming situation; then:

(a) If a new parameter is to be read, reload all the parameters in the reader, and restart by the console message GO #XPL? 20 (parameters on paper tape) or GO #XPL? 21 (parameters on cards).

(b) If a further attempt is to be made to open the file, without re-reading parameters, resume by the console message GO #XPL?
HALTED:=- OE

The compiler cannot find one of its overlays. A core dump should be taken and forwarded through the normal channels for software error.

HALTED:=- PE

An error relating to the compiler parameters has been detected. A line printer message indicates the type of error. Correct the parameters, reload all the parameters in the reader, and restart by GO #XPL? 20 (parameters on paper tape) or GO #XPL? 21 (parameters on cards).

HALTED:=- PM

The automatic post-mortem would have been entered had switch 14 not been set on. See the information on post-mortem procedures at the end of this sub-section.

The following message may be output on the console typewriter by the compilers #XPLM and #XPLX:

DISPLAY:=- FF

The compiler's post-mortem procedure is in operation but the post-mortem output will not be complete due to a failure to extend the work file. The post-mortem continues.

The following messages may be output on the line printer by the compilers #XPLF, #XPLM, #XPLX and #XPLZ:

DISC TRANSFER ERROR n UNIT x

An attempt to read from or write to a disc file has failed. x = the compiler's relative unit number of the failed file. n = the contents of bits 0 to 5 of the reply word printed as a decimal number. For the significance of the reply see the manual 'Direct Access'.

If n = 8 the remainder of the batch is abandoned; otherwise, the program being compiled is abandoned and the compiler proceeds to the next program of the batch. In either case, the SERV message is displayed on the console for the program being compiled at the time of the failure.

ERROR IN PARAMETER ABOVE

The parameter last printed does not conform to the specifications. The message HALTED:=- PE is output on the console typewriter.

FILE EXTENSION FAILED – ERROR n UNIT x

An attempt to extend the output file or (#XPLM and #XPLX only) the work file has failed. x = the compiler's relative unit number of the file concerned. n = the contents of bits 6 to 23 of the reply word printed as a decimal number. For the significance of the reply see the manual 'Direct Access'.

If p = 2 (output file) the remainder of the batch is abandoned; if p = 1 (work file) the program being compiled is abandoned and the compiler proceeds to the next program of the batch.

In either case, the SERV message is displayed on the console for the program being compiled at the time of the failure.

filename EXTENDED TO nnnn BUCKETS

This message is output at the end of the run if the file has been successfully extended in the course of the run. nnnn gives the number of buckets in the file at the end of the run.

filename NOT OPENED

A permanent file with the file name specified in a parameter cannot be found on line. The message HALTED:=- On is output on the console typewriter.

filename NOT RENAMED

The compiler is not able to rename the specified disc file.

filename *nnnnnn USED AS x

This message, where *nnnnnn is the cartridge or unit serial number and x is the compiler's relative unit number of the file, is printed when a permanent file is opened.

INVALID PARAMETER COMBINATION

Either a REN parameter is present without an accompanying OUTE or OUTF parameter; or a BIN or BINE or BINF parameter is present but the parameters which are required to accompany this parameter are not all present; or a LIBE or LIBF parameter is present without an accompanying BIN or BINE or BINF parameter. The message HALTED:=- PE is output on the console typewriter.

LAST BUCKET USED OF filename nnnn

This message is output when any permanent file which is used by the compiler is closed. nnnn gives the bucket number of the last bucket used.
NO OBJECT OUTPUT
The compiler parameters have requested the automatic consolidation facility, but OBJECT is not present in the steering information. The steering information present is acted upon, but automatic consolidation is not attempted.

NO PROGRAM LEVEL STEERING
A program is not preceded by a valid #STEER steering line (#XPLF and #XPLM); or no source segment in the subfile specified by the SUB parameter has valid steering information in its #STEER record, and no valid external steering is provided (#XPLX and #XPLZ). (Valid steering must include at least one of LIST, SHORTLIST or OBJECT.) The compiler assumes SHORTLIST, OBJECT as the program level steering information for the program, and proceeds. When the program has been compiled the COMP ER version of the end of compilation message is displayed on the console typewriter, irrespective of whether other errors were detected in the source program.

OUTPUT FILE HAS UNACCEPTABLE FORMAT
The output file specified by an APPE or APPF parameter does not have in its level 0 directory subfile a subfile description describing unused buckets (subfile type 0000).

OUTPUT FILE REQUIRED BUT NOT SPECIFIED
A source segment with OBJECT steering information, or a semi-compiled segment which is required to be copied across to the output file, has been encountered, but there was no OUTE or OUTF or APPE or APPF parameter present at the start of the run. The message HALTED:-- PE is output on the console typewriter.

STEERING BKT. nnnn
This message is printed at the end of the run if automatic consolidation has been requested. nnnn indicates the number of the bucket in the compiler's output file which contains the parameters required by the consolidator.

*TABLE EXCEEDED nnnn WORDS AT LINE xxxx
For an explanation of this message see page 20 of this chapter.

TABLE FULL #name
The combined symbol and branch ahead table has been filled and the compiler has been unable to extend its core storage space, so compilation cannot continue. The rest of the program being compiled is abandoned, the SERV message is displayed on the console, and the compiler proceeds to the next program in the batch.

TOO MANY LIB PARAMETERS
More than six LIBE and/or LIBF parameters have been read. The message HALTED:-- PE is output on the console typewriter.

TWO PARAMETERS SAME TYPE
A second parameter of a type already read (other than LIBE or LIBF) has been encountered. The message HALTED:-- PE is output on the console typewriter.

The following message may be output on the line printer by #XPLF and #XPLM:

READER FAILED HERE
There has been an Executive FAIL message in respect of the input device (the paper tape reader or card reader) and the operator has typed GO #XPL? to continue the run. If LIST or SHORTLIST is present in the steering information, this message is printed to indicate the line on which the failure occurred. The message is flagged with O and P error flags (see Chapter 9). Compilation resumes from the next line.

The following message may be output on the line printer by #XPLM and #XPLX:

SCRATCH FILE 1 NOT OPENED
A scratch file is required as a work file, but there is insufficient disc space available. The message HALTED:-- On is output on the console typewriter.

The following messages may be output on the line printer by #XPLX and #XPLZ:

EXTERNAL STEERING INVALID
The steering information contained in the SUB parameter includes neither LIST nor SHORTLIST nor OBJECT. The compiler proceeds, using the steering information contained in the file.
The input file is not in the correct format. n indicates the type of format error, thus:

- n = 1: More than 10 amendment fragments for one segment have been detected.
- n = 2: The subfile type in a level 0 subfile description is not C100 or 0000.
- n = 3: The subfile type in a level 1 subfile description is not B2P0, B2P9 or B400.
- n = 4: A subfile description of an amendment fragment is not immediately preceded by a subfile description of a source segment or of another amendment fragment.

The message HALTED:- IF is output on the console typewriter.

A subfile with the name specified in a SUB parameter cannot be found in the input file. The compiler proceeds immediately to its end of run routines.

If a compiler in this group is suspected of looping first output word 8 on the console by typing:

OU #XPL? 8

then continue with the post-mortem procedure.

The post-mortem procedure, should the compiler go ILLEGAL or be suspected of looping, is to input the console message:

GO #XPL? 29

The end of the post-mortem is indicated by the console message:

0#XPL?; HALTED:- END OF PM

The line printer output and the console log should be forwarded through the normal channels for software error.

If it ever becomes necessary to inhibit the compiler’s automatic post-mortem, this may be done by setting switch 14 on before commencing the run. Should the circumstances then arise which, were switch 14 not on, would initiate the automatic post-mortem, the compiler will halt with the console message:

0#XPL?; HALTED:- PM

In this event, an Executive core print of the compiler’s area of store should be taken and forwarded, together with the console log and printouts of the disc files used, through the normal channels for software error. Typing:

GO #XPL?

in these circumstances will cause the compiler’s post-mortem routines to be entered.
THE PLAN 2 COMPILER #XPLD

The PLAN 2 compiler, #XPLD, is designed for 8K processors with basic peripherals only, and may be used in configurations with or without a console typewriter.

Form of Source Input

The overall unit of compilation is one source program punched on eight-track paper tape or 80-column cards. A program consists of one or more segments headed by a steering line and terminated by a #FINISH directive. A segment is the basic unit of compilation and consists of PLAN statements headed by a #PROGRAM title line and terminated by a #END directive.

The steering line for this compiler is discussed on pages 34 and 39.

Object Program Output

If object program output is requested by steering information, unconsolidated semi-compiled object program is output to paper tape or cards, as determined by the setting of switch 22 (see 'Basic Peripheral Usage' below).

To obtain consolidated semi-compiled program a separate consolidation run, using #XPCA for object program on paper tape or #XPCC for object program on cards, is required.

Basic Peripheral Usage

LOAD DEVICE

The compiler may be loaded from either paper tape or cards.

INPUT DEVICE

The input device, a paper tape reader or a card reader, is allocated initially as determined by the setting of switch 23 (off for paper tape input, on for card input) and is released when either the #FINISH directive is read or there is a change in the input medium. The input medium can be changed at any time by a #SWITCH directive. When this is encountered in the input the present reader is released and one of the other type is allocated.

If a paper tape reader is used, it is disengaged after reading the steering line and after each #END directive in the source program.

PAPER TAPE PUNCH

If switch 22 is off, the compiler allocates a paper tape punch as TP0 when a program with OBJECT on its steering line is presented, and releases it when the #FINISH directive is read. If switch 22 is on, the output is on cards. It is not possible to switch output once the run has begun.

CARD PUNCH

If switch 22 is on, the compiler allocates a card punch as CP0 when a program with OBJECT on its steering line is presented, and releases it when the #FINISH directive is read. If switch 22 is off, the output is on paper tape. It is not possible to switch output once the run has begun.

LINE PRINTER

The compiler allocates a line printer as LP0 when a program with LIST or SHORTLIST on its steering line is presented, and releases it when the #FINISH directive is read.

Operating Instructions

(Lights displays and handswitch messages for machines without a console typewriter are shown in parentheses.)

1 Load the compiler into core store by an appropriate LOAD message.

2 Load the source program, preceded by its steering line, on the paper tape reader or card reader.
3 If input is on cards, input the console message:
   ON 23   (#0400  0027)

4 If OBJECT is on the steering line, and the semi-compiled object program output is to be on cards, input the console message:
   ON 22   (#0400  0026)

5 If LIST or SHORTLIST is on the steering line and a 96 print position printer is to be used, input the console message:
   ON 21   (#0400  0025)

6 Activate the compiler by the console message:
   GO 20   (#0100  0024)

7 At the end of the run the peripherals are released and the compiler halts with the console message:
   HALTED EC   (#2400  4543)

8 Either delete #XPLD or return to step 2 to compile another program.

Exception Conditions

The following messages may be output on the console typewriter:

HALTED CR   (#2400  4362)  One of these messages is output if the compiler tries to allocate a peripheral which is not available. (‘NP’ indicates a paper tape punch or a card punch, according to the setting of switch 22.) Make the required peripheral available and resume by ‘GO’.
HALTED LP   (#2400  5460)  
HALTED NP   (#2400  5660)  
HALTED TR   (#2400  6462)  

HALTED IS   (#2400  5163)  An incorrect steering line has been read. Replace it by a correct one and resume by ‘GO’.
HALTED SF   (#2400  6346)  The symbol table is full and compilation cannot continue. Rewrite or divide the segment and recompile.
HALTED VF   (#2400  6646)  The branch ahead table is full, and compilation cannot continue. Rewrite or subdivide the segment and recompile.

In the event of a parity failure on the paper tape reader, if the tape is repositioned at the beginning of the block on which the failure occurred it may be re-read by inputting the console message:

   GO 21   (#0100  0025)

If the compiler goes ILLEGAL or is suspected of looping, a core print should be taken and forwarded through the normal channels for software error.
PLAN 1 COMPILERS

PLAN 1 compilers are designed for 4K processors, and may be used in configurations with or without a console typewriter. There are two compilers available: #XLP, with input and output on paper tape or cards, and #XPLQ, with input on paper tape or cards and output on cassette tape.

Form of Source Input

The overall unit of compilation is one source program punched on eight-track paper tape or 80-column cards. A program consists of one segment headed by the #PROGRAM title line and terminated by a #END directive. The program is not preceded by a steering line; the options required when compiling are specified by console switching (see the relevant operating instructions below).

In addition to the normal features of the PLAN 1 language, the PLAN 1 compiler #XPLQ can handle SD macros referring to cassette tape only. Labels beginning with the letters BFQ are not available to the user because the labels of the buffer areas generated by the SDDEF macro commence with these letters.

Object Program Output

Unless suppressed by console switching (see the relevant operating instructions below), unconsolidated semi-compiled program is output.

#XLP outputs to paper tape or to cards, as determined by console switching. To obtain consolidated semi-compiled program a separate consolidation run, using #XPCA for object program on paper tape or #XPCC for object program on cards, is required.

#XPLQ outputs to a scratch cassette tape which is relabelled SEMICOMPTAPE with a zero retention period. The unconsolidated semi-compiled object program is written in a subfile named NAMEyyyyMMdd, where NAME is the name of the program being compiled. To obtain consolidated semi-compiled program a separate consolidation run using #XPCW is required.

Basic Peripheral Usage, #XLP

INPUT DEVICE

The compiler program is supplied in two parts, #XLP Main Binary and #XLP Overlay Binary. The input device is required to read, first the Main Binary, then the source program, then the Overlay Binary. The type of device, a paper tape reader or a card reader, is determined by console switching (see 'Operating Instructions' below). The reader is released at the end of the run. #XLP may alternatively be loaded from a different device from that used for the source program, but the two parts, Main Binary and Overlay Binary, must be loaded from the same unit.

OUTPUT PUNCH

If switch 0 is off, a paper tape punch or a card punch is allocated, as determined by the setting of switch 2. The punch is released at the end of the run.
LINE PRINTER
If switch 3 is off, a line printer is allocated and a listing of the program being compiled is produced. The line printer is released at the end of the run.

Operating Instructions, #XPLP
(Lights displays and handswitch messages for machines without a console typewriter are shown in parentheses.)

1 Load #XPLP Main Binary on the paper tape reader or card reader and read it into core store by an appropriate LOAD message.
2 Load the source program on the paper tape reader or card reader.
3 Ensure that the switches are set appropriately to the input/output requirements, in accordance with the following table:

<table>
<thead>
<tr>
<th>SWITCH 0</th>
<th>SWITCH 1</th>
<th>SWITCH 2</th>
<th>SWITCH 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>No object program output</td>
<td>Input on paper tape</td>
<td>Output on paper tape</td>
</tr>
<tr>
<td>OFF</td>
<td>Object program output required</td>
<td>Input on cards</td>
<td>Output on cards</td>
</tr>
</tbody>
</table>

4 Activate the compiler by the console message:

   GO 20             (#0100 0024)

5 When #END is read the compiler outputs the console message:

   DISPLAY OV       (#2500 5766)

   The reader required for the overlay binary is disengaged.

6 Load #XPLP Overlay Binary on the same reader as was used for #XPLP Main Binary, and allocate the reader.

7 The compiler reads in the overlay, completes its end routines, releases its peripherals, and halts with the console message:

   HALTED EC        (#2400 4543)

Basic Peripheral Usage, #XPLQ

INPUT DEVICE
The input device, a paper tape reader or a card reader, is allocated as determined by console switching (see 'Operating Instructions' below). The reader is released at the end of the run.

LINE PRINTER
If switch 2 is off, a line printer is allocated and a listing of the program being compiled is produced. The line printer is released at the end of the run.

Cassette Tape Usage #XPLQ

OBLIGATORY CASSETTE TAPES
The compiler is an overlay program loaded from the P.L.T., so this tape is retained as CT0 until the compiler is deleted.

A scratch tape is opened as CT1 and used as a work tape. It is closed and left scratch at the end of the run.

OPTIONAL CASSETTE TAPE
If switch 0 is on, no object program output is produced, so no further cassette tapes are required.
If switch 0 is off, a second scratch tape is opened as CT3 to receive the output, as described under 'Object Program Output' above.

Operating Instructions, #XPLQ
(Lights displays and handswitch messages for machines without a console typewriter are shown in parentheses.)
1 Set up the P.L.T. on station 0, and scratch cassettes with write permit buttons on station 1 and (if unconsolidated semi-compiled object program is required) station 3.

2 Load the compiler into core store by an appropriate FIND message.

3 Load the source program on the paper tape reader or card reader.

4 Ensure that the switches are set appropriately to the input/output requirements, in accordance with the following table:

<table>
<thead>
<tr>
<th>SWITCH 0</th>
<th>SWITCH 1</th>
<th>SWITCH 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>No object program output</td>
<td>Input on paper tape</td>
</tr>
<tr>
<td>OFF</td>
<td>Object program output required</td>
<td>Input on cards</td>
</tr>
</tbody>
</table>

5 Activate the compiler by the console message:

GO 20  (#0100 0024)

6 At the end of the run the compiler closes the cassette tapes, releases the basic peripherals and halts with the console message:

HALTED EC  (#2400 4543)

Note: if a further compilation is required the compiler must be reloaded.

Exception Conditions, #XPLP and #XPLQ

The following messages may be output on the console typewriter:

DISPLAY RD  (#2500 6244)  An error has been detected on the input medium, for example, an unacceptable character, or more than 128 characters on a paper tape line. The reader is disengaged. The faulty line is error flagged on the compilation list if a listing is being produced. The operator should mark the error on the input medium, and allocate the reader to resume compilation from the next line.

DISPLAY TF  (#2500 6446)  The combined symbol and branch ahead table is full. Compilation continues, but all further symbol definitions are ignored and error flagged unless either a #OUST directive or an evaluation of a branch ahead makes room in the table.

HALTED EF  (#2400 4546)  The program being compiled has attempted to allocate more than 4096 words of core store. Abandon the run and rewrite the program.

HALTED Pn  (#2400 600n)  The compiler has tried to allocate a peripheral which is not available. The peripheral type is indicated by n as follows:

n = 0: Paper tape reader.
n = 1: Paper tape punch.
n = 2: Line printer.
n = 3: Card reader.
n = 4: Card punch.

Make the required device available and resume by 'GO'.

The following console messages may be output by #XPLQ only:

HALTED 02  (#2400 0002)  Any of these messages indicates an error condition in the compiler. A core print and a printout of CT1 should be taken and forwarded through the normal channels for software error.

HALTED 04  (#2400 0004)

HALTED CD  (#2400 4344)
If the compiler goes ILLEGAL or is suspected of looping, a core print and, if applicable, a printout of CTJ should be taken and forwarded through the normal channels for software error.

CONSOLIDATION
This section is intended to augment the information on the consolidation process given earlier in this chapter.

Compilers with Integral Consolidators
These comprise the compilers with output on magnetic tape or cassette tape, with the exception of #XPLR.

NON-OVERLAY PROGRAMS
Consolidation is effected in one pass of the semi-compiled work tape, plus one pass of the program library tape if necessary. There is then a copying pass from the work tape to the final output tape.

If the program to be compiled uses no library subroutines, time is sometimes wasted doing a library search because of a mis-punch somewhere in the source program. To avoid this possibility, a #COMPLETE directive should be included in the source immediately before the last #END. This will prevent the compiler searching the library tape even if there are unsatisfied cues. Any unsatisfied cues will be dealt with as described on page 3 and under 'Exception Conditions'.

OVERLAY PROGRAMS
The segments of an overlay program must be presented to the consolidation process in the following order:

1. The steering segment first.
2. Overlay segments next, the segments being arranged within overlay units within overlay areas. Overlay areas, and overlay units within overlay areas, must be in rising sequence. Segments within overlay units must be in the sequence in which they were specified under the #OVERLAY directive.
3. Permanent segments last.

However, it is not essential for the segments to be input to the compiler in this strict sequence for, provided that the steering segment is input first, the compiler will sort the segments into correct order. Unless a #ORDER directive is present in the steering segment, the compiler will assume that the segments input are not in the required sequence and will proceed to sort them. If the segments are in the correct order for presentation to the consolidation process, therefore, time will be saved if a #ORDER directive (see Chapter 6) is included in the steering segment. Consolidation is then carried out as for a non-overlay program.

If a #ORDER directive is not present in the steering segment, the number of tape passes required during the consolidation of an overlay program is as follows: one pass down the work tape, plus one pass down the program library tape if necessary, plus as many additional passes down the work tape as are required to get the segments into order (with a minimum of one). With the compiler #XPLS a copy pass down the work tape is also required.

The pass of the program library tape is generally obligatory because the program will require at least the overlay package.

If a #ORDER directive is present in the steering segment the G.P.L. will precede the semi-compiled object program on the output tape. If a #ORDER directive is not present in the steering segment then, except when the compiler #XPLS is used, the G.P.L. will follow the semi-compiled object program on the output tape, and two passes down this tape will be needed during the first phase of loading the program.

The directive #COMPLETE can be used with overlay programs as with non-overlay programs, but it should be borne in mind that this will inhibit the inclusion of the overlay package from the library tape; it should only be used, therefore, where the program is supplying its own overlay package, or where a partial test run of permanent segments only is required.

Compilers without Integral Consolidators
These comprise the compilers with output on disc, plus the compilers #XPLE, #XPLR and the PLAN 2 and PLAN 1 compilers. These compilers produce unconsolidated semi-compiled output, and so must be followed by a separate consolidator program run before the object programs can be loaded. The consolidators required, and, for disc output compilers, a method of calling the consolidator automatically by means of compiler parameters, have been mentioned earlier in this chapter; the table below summarises the consolidators appropriate for use after the various compilers.
<table>
<thead>
<tr>
<th>Compiler</th>
<th>Semi-Compiled Output on</th>
<th>Consolidator</th>
<th>Consolidated Output on</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#XPLE</td>
<td>Paper tape</td>
<td>#XPCA</td>
<td>Paper tape</td>
</tr>
<tr>
<td></td>
<td>Cards</td>
<td>#XPCC</td>
<td>Cards</td>
</tr>
<tr>
<td>#XPLR</td>
<td>Cassette tape</td>
<td>#XPCX</td>
<td>Cassette tape</td>
</tr>
<tr>
<td>#XPLM</td>
<td>Disc</td>
<td>#XPCL</td>
<td>Disc</td>
</tr>
<tr>
<td>#XPLX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#XPLF</td>
<td>Disc</td>
<td>#XPCK</td>
<td>Disc</td>
</tr>
<tr>
<td>#XPLZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLAN 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#XPLD</td>
<td>Paper tape</td>
<td>#XPCA</td>
<td>Paper tape</td>
</tr>
<tr>
<td></td>
<td>Cards</td>
<td>#XPCC</td>
<td>Cards</td>
</tr>
<tr>
<td>PLAN 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#XPLP</td>
<td>Paper tape</td>
<td>#XPCA</td>
<td>Paper tape</td>
</tr>
<tr>
<td></td>
<td>Cards</td>
<td>#XPCC</td>
<td>Cards</td>
</tr>
<tr>
<td>#XPLQ</td>
<td>Cassette tape</td>
<td>#XPCW</td>
<td>Cassette tape</td>
</tr>
</tbody>
</table>

Details of the consolidator programs #XPCA and #XPCC may be found in the manual Basic and Commercial Utilities, of #XPCL and #XPCK in the manual Compiling Systems, and of #XPCW and #XPCX in the manual Cassette Tape Routines.

**LOADING OBJECT PROGRAMS FROM MAGNETIC TAPE AND CASSETTE TAPE**

When a batch has been compiled and consolidated the output tape contains the search program, in binary, and one or more consolidated programs in G.P.L. load form (see pages 4 to 6 of this chapter). In the absence of compiler parameters, the tape will be labelled PROGRAM TAPE if a magnetic tape, or PROGRAM XPKP if a cassette tape.

**Non-overlay Programs**

A program can be loaded from the tape by means of the console directive FIND. In the case of non-overlay programs, the message

**HALTED LD**

from the non-overlay loader (XPO1 for magnetic tape or XPP1 for cassette tape) indicates that the object program is ready to run.

It is possible to specify by a steering line for the overlay loader (XPO6 for magnetic tape or XPP6 for cassette tape) to be given to a non-overlay program. When the program is subsequently loaded this will result in the writing of a tape labelled PROGRAM NAME (where NAME is the name of the program being loaded) containing the program in binary form, as described below.

**Overlay Programs**

Loading of overlay programs from the compiler or consolidator output tape is done in two phases. In the first phase the program is located by means of the console directive FIND, and the overlay loader (XPO6 for magnetic tape or XPP6 for cassette tape) is brought into core. The overlay loader then opens a scratch tape, relabels it PROGRAM NAME (where NAME is the name of the program being loaded) and writes to it in binary form firstly the overlays and then the permanent program. The binary program is written in a form suitable for chaining. At the end of this phase the loader deletes itself with the message:

**DELETED FIND #NAME**

The program is then loaded into core from the binary tape, and the message

**HALTED LD**

indicates that the object program is ready to run.
Chapter 8 PLAN 4

PLAN 4 offers all the facilities of PLAN 3, with the following additional features:

1. Features designed to facilitate the writing and compilation of programs which are intended to be run in extended mode or/and extended branch mode. These features include two further directive statements, #PMODE and #HMODE, an expanded form of the #CUE directive, expanded forms of the #PROGRAM directive, and four further macro-instructions.

2. The #ELASTIC directive, which enables the user to designate a particular upper common variable data area to be placed by the consolidator at the top of the program's storage area; thus if it becomes necessary to allocate additional core store, it is the designated area which is extended.

3. Provision for the user to specify a segment which is to be treated by the consolidator as satisfying all otherwise unsatisfied cues. This is done by means of a #ERRORSEG directive and a further expanded form of the #PROGRAM directive.

The various directive statements mentioned above are described in detail in Chapter 6.

The four further macro-instructions available in PLAN 4 are:

- BUX X,Xm N
- BDX Xc,Xm N
- BCHX Xc,Xm N
- LDCM X,Xm N(M)

These are provided for ease of handling the two-word counter-modifiers which may be required when in extended data mode; they are described individually in Chapter 4.

The user may indicate to the compiler in which address mode or modes and in which branch mode or modes each segment is designed to operate; may specify what compatibility checks are to be performed on the address modes and branch modes of the segments comprising a program; may specify the address mode in which the program is initially to operate after loading; and may specify the overall branch mode in which the program is to operate. Although the address mode may be changed whenever required, a PLAN 4 program must operate in one branch mode throughout.

A PLAN 4 program must commence with a steering segment, which must contain a #PMODE directive. The PLAN 4 steering segment must have a program name in the #PROGRAM directive. If a program name is not present then the mode will be set to 15AM and DBM.

**COMPILER TABLE SIZES**

The PLAN 4 compilers use tables during the compilation process to hold details of the symbolic identifiers, macro-instructions, literals, branches ahead and cues that are encountered. The sizes of these tables are limited, but the compiler will, if the need arises, attempt to expand the combined symbol and branch ahead table by increasing the amount of core store allocated to the compiler. If the attempt is unsuccessful the program currently being compiled is abandoned, and the reason for the abandonment is printed on the line printer. #XPLN will attempt to expand the consolidated cue list in a similar manner if the need arises. A larger symbol and branch ahead table may be provided by specifying a suitable storage parameter in the FIND or LOAD message when the compiler is loaded.

**Combined Symbol and Branch Ahead Table**

The size of the combined symbol and branch ahead table in PLAN 4 compilers designed for 32K processors is 1600 words.

Every symbol occupies two words if #XPLN and three words if #XPLT plus the number of words required to contain its name in character form. Segment names, peripheral names, common block names and data names within common areas require a further word.
Every branch ahead occupies two words. A further one or two words are required if the forward reference occurs in a common area.

The sizes quoted above for this table are the 'official' sizes which have been allowed in the compilers. The actual sizes of the table are greater than the official sizes, but this may not be a permanent feature; additionally, compilers will attempt to extend their table size if necessary. If the amount of table space used in the course of a compilation exceeds the 'official' table size and LIST, SHORTLIST, MAP or FULLLIST appears in the steering information, then a warning message is output on the segment's error line, to the right of the error code information. The format of this message is:

*TABLE EXCEEDED 1600 WORDS AT LINE XXXX

where XXXX is the number of the line which caused the 'official' table size to be exceeded. This message is only a warning, in case the program should subsequently be recompiled with a smaller available table space; it does not mean that the compilation is abortive.

Literal Table

The size of the literal table in all PLAN 4 compilers is 75 words.

Every literal occupies one word in the table. One extra word is used (and inserted in the object program) if the literal makes reference to a word in a common area. If the table becomes full during compilation and a listing is being produced, the compilation is interrupted and all 75 literals are printed out. Whether the compilation is being listed or not the literal table is zeroized and is ready to be filled again.

Macro-Instruction Table

The size of the macro-instruction table in #XPLN is 2500 words and in #XPLT is 1800 words.

PLAN macro-instructions take up about 250 words. Every macro-instruction takes up approximately three words plus a further three words for each basic instruction that is required. For example, BXGE takes up nine words; three in itself plus three for TXL and three for BCC. Apart from PLAN macro-instructions, there is room in the #XPLN table for about 200 average-sized, user-defined macro-instructions and in the #XPLT table for about 150.

Consolidated Cue List (#XPLN only)

The size of the consolidated cue list in #XPLN is 1000 words.

Each cue name occupies three words plus the number of words required to contain its name in character form.

Map/Fulllist Table (#XPLN only)

This table re-uses some of the locations previously used for the combined symbol and branch ahead table and for the consolidated cue list. Information for the preparation of a map and fulllist, if either of those facilities is requested by steering information, is written to magnetic tape during compilation and consolidation and is read back into core store when consolidation has been completed. If at that time it is found that there is insufficient space available to read in all the necessary information, the map and/or fulllist is abandoned; object program output, however, will already have been achieved, and although the 'SERV' message is output on the console (see 'Operating Instructions for #XPLN, page 14) the program may be run if required.

The PLAN 4 compilers are #XPLN, described below, and #XPLT, a description of which commences on page 16.

THE PLAN 4 COMPILER #XPLN

#XPLN is a batch compiler on magnetic tape for machines of 32K and above. It provides for the compilation of extended data mode programs, but not for the compilation of extended branch mode programs. It implements all the other features of the PLAN 4 language. Both source and semi-compiled input may be accepted, from magnetic tape, paper tape and/or cards; the semi-compiled object programs are output to magnetic tape. A file of unconsolidated semi-compiled segments may also be output to magnetic tape.
The hardware requirements for #XPLN are:

- A central processor with not less than 32K words.
- One paper tape reader and/or one card reader.
- One line printer (optional).
- One magnetic tape deck.
- Up to eight further magnetic tape decks (optional).

The use of these peripherals is described in the subsections which follow.

The compiler is held on the P.L.T., with a priority as supplied of 93.

Besides all the normal facilities of the PLAN 4 language other than extended branch mode features, #XPLN can generate calling sequences to the following compact mode software packages:

- MONITOR.
- Input/Output Generator.
- Magnetic Tape Housekeeping system.
- Storage Device Housekeeping systems (15-bit address mode versions).
- Overlay system.
- Dump and Restart package.

#XPLN can also generate calling sequences to the following packages which are designed to work both in 15-bit address mode and in 22-bit address mode:

- MONITORX
- Storage Device Housekeeping systems (address mode-compatible versions).

Character Peripheral Usage

An input peripheral is required to read control parameters (described later) and may also be used to read PLAN source and/or semi-compiled segments. The required peripheral, a paper tape reader or a card reader, is determined initially by the choice of entry point used (see 'Operating Instructions' on page 14), and is released when either all the input has been read or there is a change of character peripheral input medium. The character peripheral input medium for parameters can be changed, by a #SWITCH directive, at any time after the first two paper tape lines or cards have been read. When a #SWITCH directive is encountered in the input data the current reader is released and one of the other type is allocated. PLAN source or semi-compiled segments may be read from the reader that currently holds the parameters, or from a reader of the other type if so specified in a parameter. In the latter case the required reader is allocated when the parameter is read and is released after the number of segments specified by the parameter has been read.

A line printer is allocated as LPO when the compiler is entered, and is released at the end of the run. In addition to providing any compiler listings or store maps that may be requested by steering information, the line printer is used for certain error messages (see page 12) and to print any parameters which contain errors. Parameters other than those in error are not printed.

Magnetic Tape Usage

The compiler is an overlay program loaded from the P.L.T., so this tape is retained as MTO until the program is deleted.

INPUT TAPES

Up to six magnetic tape decks may optionally be used for input, controlled by parameters as described later in this chapter. However, a maximum of six named tapes may be specified at any one time, to cover both input and output requirements; so if named tapes are required for output, the number of decks that can be used for input is correspondingly reduced.

Output tapes produced by the magnetic tape COSY and DISC COSY editor programs (see Chapters 10 and 11), those produced by the GEORGE editor, #KXYA using the SNB instruction, and those produced by PLAN compilers are suitable for use as input to #XPLN. PLAN source and/or semi-compiled segments may be input from such magnetic tapes under the selective control of parameters in the paper tape/card input file.
If magnetic tapes written by programs other than those already mentioned are to be used as input, the following points should be noted.

All magnetic tapes used for input must be single reel files in 1900 Series standard subfile format. Blocks may be of variable length, with a maximum size of 512 words. Records may also be of variable length, with a maximum size of 21 words of source program or 18 words of semi-compiled program. Each simple subfile must contain either only PLAN source records or only semi-compiled records.

A segment, whether in source or semi-compiled form, must occupy a number of consecutive records. A segment must start at the beginning of a new block. If a source segment starts with a #STEER record (as on tapes produced by the magnetic tape COSV editor programs) #XPLN will skip over this record. Semi-compiled segments may be selected from files of unconsolidated semi-compiled segments, or from files containing consolidated programs in G.P.L. load form.

MAGNETIC TAPE FILE NUMBERS AND UNIT NUMBERS

It will be seen in the description of the control parameters, later in this chapter, that named tapes used for input or output are given a relative file number, X, in the range 1 to 6. This file number is distinct from, and is not the same as, the program's relative unit number of the tape deck holding the file. Where a tape deck is holding a specified named tape, that deck is allocated as MT(16-X) when the tape is opened.

If a tape is closed, the file number X specified for that tape becomes available for re-use; thus, although a maximum of six tapes may be specified to be open at any one time, more than six named tapes may be opened in the course of a run. Any tape may be closed specifically by a CLOSE parameter. Certain tapes may be closed automatically when a parameter opening another tape of a like kind is read; see the definitions of the individual parameters in the following pages. Any tapes remaining open when the run terminator is read are closed before the program halts. Trailer labels are written by #XPLN wherever appropriate when files are closed.

OUTPUT TAPE

If consolidated semi-compiled object programs are required, two of the optional magnetic tape decks are needed. The object program output tape may be a named tape or a scratch tape (i.e. one with an expired retention period; a parametrically opened scratch tape, with a file name specified as SCRATCH TAPE, is regarded as a named tape). If a named tape is specified it is opened as MT(16-X). If a scratch tape is opened, it is opened as MT3. A work tape is also opened, as MT2, and is left scratch at the end of the run.

If a named tape is specified for the object program output tape, it may be specified to be opened as a new composite file to receive a batch of programs, or as an existing composite file to which further programs are to be added, or as a simple file to receive a single program. (See the OWFA, APPA and OWFB parameters, below.) If either of the first two of these is specified, the output tape is suitable for subsequent use as input to #XPLN; if the last, simple file, form is specified, it is not suitable for subsequent use as input to #XPLN. Only one object program output tape may be open at any one time; but the opening of a new object program output tape automatically closes any other named object program output tape that is open at the time, so a named object program output tape may be closed and a new one opened whenever necessary. (Alternatively, a CLOSE parameter may be used specifically to close any named tape during the run.)

If a scratch tape is opened as the object program output tape, it is opened as a composite file, and is left open until the end of the run. Any OWFA, APPA or OWFB parameters read after the scratch tape is opened will be ignored.

Whether or not an object program tape is called for, an output file of unconsolidated semi-compiled segments may be produced. If this facility is required, a named tape must be specified, which will be opened as MT(16-X). The tape will be written in 1900 Series standard subfile format, and will be suitable for subsequent use as input to #XPLN. The semi-compiled segments may either be written to a newly created file (by means of the OWFC parameter) or be added to an existing file (by means of the APPC parameter.)

Character Peripheral Input File

The input file on paper tape and/or cards contains parameters to control the run. These are described in detail in the next subsection of this chapter. Interspersed among the control parameters there may be segments of program in PLAN source and/or semi-compiled form.
A run of #XPLN comprises a number of jobs. Each job consists of the compilation of a single program. The segments of the program may be gathered together in source or in semi-compiled form from paper tape, cards and/or a number of different magnetic tapes. Each job is defined by a series of parameters, commencing with a JOB parameter and ending with a job terminator; between these are the various parameters controlling the input and output, together with any input segments which are on paper tape or cards. Following the last job of the batch is the file terminator which terminates the run.

Any magnetic tapes opened in the course of a job are not automatically closed at the end of the job, but will remain open until specifically closed by a CLOSE parameter, or until automatically closed by a parameter opening another tape of like kind as defined in the individual parameter specifications below, or until the file terminator is read. Thus a number of jobs can use the same input and/or output files, but any job can specify different input and/or output files if necessary; any files left open at the end of the run are automatically closed.

Control Parameters

The control parameters have a format similar to that of PLAN instruction statements, i.e., they use the same label, operation, accumulator and operand fields. Each parameter line is punched on a separate paper tape line or a separate card. In the case of paper tape, each line is terminated by a newline character and has a maximum permitted size of 128 internal characters; beta shift characters and delta shift characters other than $\$, $\uparrow$, $\downarrow$, a newline and horizontal tabulation are not accepted. If horizontal tabulation characters are used, they will be expanded to the field format of PLAN source cards.

The control parameters are as follows:

1. To specify a job.

   Label  Operation  Accumulator  Operand
   *JOB NAME,XXXXXXX
   where NAME = four alphanumeric characters, of which the first must be alphabetic. This specifies the name which is to be inserted as the program name in the segment leader of each semi-compiled segment output until the next job terminator parameter is read; and hence specifies also the name of the consolidated program, if consolidated program output is requested. NAME must not be omitted, for any program names present in the input are ignored for these purposes.

   XXXXXXX = up to eight characters which, if present, will be displayed on the console when the job is started, in the message 'START OF JOB NAME, XXXXXXX'. They will be written to the program's request slip and may be used for any purpose as required. They may be omitted, in which case the preceding comma should also be omitted.

2. To open a read file.

   Label  Operation  Accumulator  Operand
   ORF X FILENAME, FGN/TSN
   where X = a decimal number in the range 1 to 6. This becomes the program's file number for the specified tape, for as long as the tape remains open. It may be used to identify the file in subsequent REWIND, CLOSE, PLAN or S/C parameters. It must be unique among file numbers associated with files currently open, but may have been used earlier in association with files now closed.

   FILENAME = the 12-character file name of the tape which is to be opened. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

   FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted, in which case the preceding comma may be omitted. If it is omitted or expressed as zero the file generation number will not be checked.

   TSN = the tape serial number, in octal, in the range octal 0 to octal 777777. It may be punched with or without a preceding * or # sign. It may be omitted, in which case the preceding solidus should also be omitted. If it is omitted or expressed as zero the tape serial number will not be checked.

   The specified file will be opened for input and left positioned immediately after the header label.
The ORF parameter will automatically close any other read file which is currently open and which has the same value for X; but other read files with different values for X may be concurrently open.

3 To open a write file for batched object program output.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWFA</td>
<td>X</td>
<td></td>
<td>FILENAME, FGN/TSN</td>
</tr>
</tbody>
</table>

where X, FILENAME, FGN and TSN are as described for the ORF parameter above, except that X may be used to identify the file in subsequent RWH or CLOSE parameters only.

The specified file will be opened for output and left positioned immediately after the header label. It will be renamed PROGRAM TAPE, with a file generation number of zero and a retention period of 4095, unless the OWFA parameter is immediately followed by a RWH parameter. It will receive all object programs subsequently output until it is closed by a subsequent OWFA, APPA or OWFB parameter, or until the end of the run.

The OWFA parameter will automatically close any other tape which is currently open for object program output.

4 To open and add to an existing file of batched object programs.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPA</td>
<td>X</td>
<td></td>
<td>FILENAME, FGN/TSN</td>
</tr>
</tbody>
</table>

where X, FILENAME, FGN and TSN are as described for the ORF parameter above, except that X may be used to identify the file in a subsequent CLOSE parameter only.

The specified file, which must be composite, will be opened and left positioned immediately before the trailer label. Its file name, generation number and retention period will remain unchanged. It will receive all object programs subsequently output until it is closed by a CLOSE parameter or by a subsequent APPA, OWFA or OWFB parameter, or until the end of the run.

The APPA parameter will automatically close any other tape which is currently open for object program output.

5 To open a write file for single program output.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWFB</td>
<td>X</td>
<td></td>
<td>FILENAME, FGN/TSN</td>
</tr>
</tbody>
</table>

where X, FILENAME, FGN and TSN are as described for the ORF parameter above, except that X may be used to identify the file in a subsequent CLOSE parameter only.

The specified file will be opened for output. It will be renamed PROGRAM NAME, where NAME is the program name specified in the preceding *JOB parameter, and will have a generation number of zero and a retention period of 4095. The object program will be written to it as a simple file.

The OWFB parameter should be used within the job in which the tape it specifies is required, and the specified tape should not be closed by a CLOSE parameter.

The OWFB parameter will automatically close any other tape which is currently open for object program output.

6 To open a write file for unconsolidated semi-compilled output.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWFC</td>
<td>X</td>
<td></td>
<td>FILENAME, FGN/TSN</td>
</tr>
</tbody>
</table>

where X, FILENAME, FGN and TSN are as described for the ORF parameter above, except that X may be used to identify the file in subsequent RWH, WSF, SFEND or CLOSE parameters.

The specified file will be opened for output and left positioned immediately after the header label. Its file name and generation number will not be changed unless the OWFC parameter is immediately followed by an RWH parameter. Simple subfiles may be written to this tape, a subfile being opened by a WSF parameter and closed by an SFEND parameter. These two parameters may be used to control the output to the tape.

If a subfile is opened on the tape, it will receive a copy of all semi-compilled segments subsequently output until the subfile is closed, irrespective of whether such segments were input in source or semi-compilled form, and irrespective of whether they are output in response to an OBJECT or to a CONSOLIDATE steering line (see the subsection 'Steering Information'). If a subfile is closed, no further output will be written to the tape until a new subfile is opened.

The OWFC parameter will automatically close any other tape which is currently open for unconsolidated semi-compilled output; but an object program tape may be concurrently open.
To open and add to an existing file of unconsolidated semi-compiled segments.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPC</td>
<td>X</td>
<td>FILENAME, FGN/TSN</td>
<td></td>
</tr>
</tbody>
</table>

where X, FILENAME, FGN and TSN are as described for the ORF parameter above, except that X may be used to identify the file in subsequent WSF, SFEND or CLOSE parameters.

The specified file, which must be a composite file, will be opened for output and left positioned immediately before the trailer label. Its file name and generation number will remain unchanged.

The APPC parameter will automatically close any other tape which is currently open for unconsolidated semi-compiled output; but an object program tape may be concurrently open.

To re-write a header label on an output file.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWH</td>
<td>X</td>
<td>FILENAME, FGN, RET</td>
<td></td>
</tr>
</tbody>
</table>

where X = the program's file number of the tape to be relabelled, as specified in the preceding OWFA or OWFC parameter.

FILENAME = the 12-character file name to be written to the specified output tape. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607, to be written to the specified output tape. It may be omitted. If it is omitted the tape will be given a file generation number of zero.

RET = the number of days retention period to be given to the specified output tape, in decimal, in the range 1 to 4095. It may be omitted, in which case the preceding comma may also be omitted. If it is omitted or expressed as zero the tape will be given a retention period of 4095.

If FGN is omitted but RET is present, the two commas must be present. If FGN and RET are both omitted, the two commas may be omitted.

The program will act upon this parameter only if it immediately follows an OWFA or an OWFC parameter and its X field agrees with that of the OWFA or OWFC parameter; if these two conditions are not satisfied, the parameter is ignored. If the conditions are satisfied, the specified output tape will be relabelled in the manner indicated.

Where the RWH parameter follows an OWFA parameter, if the first eight characters of FILENAME are not PROGRAM, Executive will not be able to Find programs from the tape. The last four characters of FILENAME will be the name given to the search program which is inserted as the first program on the tape. The name thus given to the search program should not be the same as the name of any object program which is to be found from the tape.

To write a subfile sentinel on the unconsolidated semi-compiled output file.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSF</td>
<td>X</td>
<td>SUBFILENAME</td>
<td></td>
</tr>
</tbody>
</table>

where X = the program's file number of a tape which was opened by an OWFC or an APPC parameter.

SUBFILENAME = the name to be given to the new subfile. It may be up to 12 characters long, of which the first must be alphabetic, and the rest may be alphanumeric or spaces or minus signs. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

A simple subfile sentinel with the specified subfile name will be written to the tape which is identified by X. The subfile should be closed by an SFEND parameter before the end of the run, or before another WSF parameter is used, or before a CLOSE parameter is used with reference to the same tape. However, should a file terminator, a further WSF parameter or a relevant CLOSE
parameter be read while the subfile is still open, such a parameter will be accepted and a preceding SFEND parameter implied.

The use of this parameter is further discussed under the OWFC parameter, above.

10 To write an end-of-subfile sentinel on the unconsolidated semi-compiled output tape.

Label Operation Accumulator Operand
SFEND X

where
X = the program's file number of a tape which was opened by an OWFC or an APPC parameter. It should be the same as the file number specified in the last preceding WSF parameter.

An end-of-subfile sentinel will be written to the tape which is indicated by X. The parameter forms a pair with the WSF parameter and should be used after all the segments which are to be included in the particular subfile have been output (as specified by PLAN or S/C parameters, and relevant steering information), or when subsequent segments output to the object program tape are not required to be written also to the unconsolidated semi-compiled output file. See further under the OWFC and WSF parameters, above.

11 To rewind an input file.

Label Operation Accumulator Operand
REWIND X

where
X = the program's file number of a tape which was opened by an ORF parameter.

The tape which is identified by X will be rewound and then positioned immediately after the header label. This parameter should be used when further segments are required from an input tape, but such segments are to be found in subfiles which occur on the tape earlier than the subfile containing the last segments read from it.

If the tape identified by X is one opened for writing, an error indication will be given, and the repositioning will not take place.

12 To close a file.

Label Operation Accumulator Operand
CLOSE X

where
X = the program's file number of the tape which is to be closed.

The tape which is identified by X will be closed. If the tape is an output file, the appropriate trailer label will be written, and if it is an unconsolidated semi-compiled output file, and the last subfile has not been closed by an SFEND parameter, then an end-of-subfile sentinel will be written before the trailer label.

The closing of the file frees the identifier X, so that another file may be opened using the particular value of the identifier.

13 To provide program level steering information.

Label Operation Accumulator Operand
ASTEER Steering information

where Steering information = LIST or SHORTLIST and/or MONITOR and/or OBJECT and/or CONSOLIDATE and/or MAP and/or FULLIST and/or L.AP(OFF) and/or FULLIST(OFF) and/or BINARY separated by commas.

For the significance of these items see the next subsection of this chapter, 'Steering Information'.

This parameter provides the steering line for all subsequent source segments of the program forming the current job, unless it is overridden for specific segments by a segment level steering parameter (BSTEER), or until a further ASTEER parameter is read. It is cancelled by the job terminator. If a further ASTEER parameter is encountered within the same job, the later one's steering information replaces that specified by the earlier one. As many ASTEER parameters may be used within a job as may be required.

Steering information should be provided for each job.
To provide segment level steering information.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
<th>Steering information</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSTEER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where Steering information is as described under ASTEER.

This parameter must be immediately followed by a PLAN parameter specifying the input of source segments. It provides the steering line for the segments specified in the PLAN parameter that follows immediately, and for those segments replaces any program level steering information. Thereafter steering reverts to that specified in the previously applying ASTEER parameter.

To read PLAN source segments from magnetic tape.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN</td>
<td>X</td>
<td>SUBFILENAME,SEGMENTNAME,(n)</td>
<td></td>
</tr>
</tbody>
</table>

where

\(X = \) the program's file number of a tape which has been opened by an ORF parameter.

\(\text{SUBFILENAME} = \) the 12-character name of the simple subfile containing the PLAN source segments to be compiled, which is to be sought on the tape identified by X. If SUBFILENAME is punched with fewer than 12 characters, the program will supply spaces at the right-hand end to complete the 12 characters.

\(\text{SEGMENTNAME} = \) the name of the first segment to be compiled from the specified subfile under the control of this parameter. It may be omitted, in which case the two commas and \(n\) should also be omitted. If it is omitted, all the segments in the specified subfile will be compiled, SEGMENTNAME may be up to 11 characters long. If it is punched with fewer than 11 characters the program will supply spaces at the right-hand end to complete the 11 characters. In the case of SEGMENTNAME, the program will remove any internal spaces, and disregard them in the count of 11 characters.

\(n = \) the number of successive segments to be compiled, starting with the one named; in decimal, in the range 1 to 511. It may be omitted, in which case the preceding comma may also be omitted. If it is omitted, \(n\) will be given the value 1, unless SEGMENTNAME is also omitted, in which case all the segments in the specified subfile will be compiled.

The specified segments will be compiled in accordance with the steering information contained in the last preceding ASTEER parameter or the immediately preceding BSTEER parameter, as the case may be. An error indication will be given if an end-of-subfile sentinel is encountered before the \(n\) segments specified have been read.

Note: #:XPLN works at the simple subfile level; the subfile names of composite subfile sentinels are not examined. The entire file will be searched for the specified subfile. If the specified subfile occurs earlier on the input tape than the subfile containing the last segments read from the tape, time will be saved if a REWIND parameter is used before the PLAN parameter.

To read PLAN source segments from paper tape or cards.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN</td>
<td>TR,(n)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or PLAN</td>
<td>CR,(n)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where

\(n = \) the number of PLAN source segments to be read from the specified reader; in decimal, in the range 1 to 511. It may be omitted, in which cases the preceding comma may also be omitted. If \(n\) is omitted, it will be given the value 1.

The specified number of PLAN source segments will be read from the paper tape reader or the card reader respectively, and compiled in accordance with the steering information contained in the last preceding ASTEER parameter or the immediately preceding BSTEER parameter, as the case may be. If the medium from which the segments are to be read is the same as the medium currently being used for the parameters, then the segments should immediately follow the PLAN parameter in the paper tape/card input file, and will be read from the same reader. If the medium from which the
segments are to be read is the alternative one to that currently being used for the parameters, then
a reader of the type specified in the PLAN parameter will be allocated as soon as the parameter
is read, and will be released when the specified number of segments has been read. #SWITCH
directives are not required for these purposes.

Each source segment should end with #END. If #FINISH directives are included, #XPLN will
ignore them.

17 To read semi-compiled segments from magnetic tape.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/C</td>
<td>X</td>
<td>SUBFILENAME,SEGMENTNAME,n</td>
<td></td>
</tr>
</tbody>
</table>

where X, SUBFILENAME, SEGMENTNAME and n are as described for the equivalent PLAN para-
meter, except that instead of PLAN source, semi-compiled segments are specified.

The specified semi-compiled segments will be consolidated and/or written to the unconsolidated
semi-compiled output file, depending upon which output tapes are currently open and what steering
information currently applies. An error indication will be given if an end-of-subfile sentinel is
encountered before the n segments specified have been read.

Notes:

(a) Overlay programs must have the steering segment in PLAN source form, in order to be
consolidated correctly.

(b) If the subfile specified contains consolidated semi-compiled program, and all the segments from
the subfile are required, the full form of the parameter should be used, with n expressed as a
number larger than the number of segments in the subfile. The form of the parameter with
SUBFILENAME as the only item in the operand field should not be used for consolidated program
subfiles.

(c) If the current steering information includes CONSOLIDATE, with or without OBJECT, then

(i) consolidated semi-compiled output will be written to the tape that was opened by the last
OWFA, APPA or OWFB parameter, if such a tape is currently open. If no OWFA, APPA or
OWFB tape is currently open, then a scratch tape will be opened to receive the consolidated
semi-compiled segments.

(ii) unconsolidated semi-compiled segments will be written to the work tape, and to the tape
opened by the last OWFC or APPC parameter, if such a tape is currently open and has a
subfile currently open.

(d) If the current steering information includes OBJECT but not CONSOLIDATE, then unconsolidated
semi-compiled segments will be written to the tape opened by the last OWFC or APPC para-
meter, if such a tape is currently open and has a subfile currently open.

18 To read semi-compiled segments from paper tape or cards.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/C</td>
<td>TR,n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/C</td>
<td>CR,n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where n is as described for the equivalent PLAN parameter, except that it refers to semi-compiled
segments.

The specified number of semi-compiled segments will be read from the paper tape reader or the
card reader respectively, and will be consolidated and/or written to the unconsolidated semi-
compiled output file, depending upon which output tapes are currently open and what steering
information currently applies. See notes (a), (c) and (d) under the magnetic tape version of the
S/C parameter.

The input medium and the reader are handled in the same way as with the equivalent PLAN parameter.

19 To switch the reading of parameters from paper tape to cards or vice versa.

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Accumulator</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#SWITCH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The current reader is released and one of the other type is allocated. Note that #SWITCH is only required for switching the input medium of the main paper tape/card input file, i.e., that containing the parameters, and which may or may not also contain input segments. The allocation and release of a reader specified in a PLAN or S/C parameter, where this differs from the reader currently in use for the file containing the parameters, do not require #SWITCH parameters.

20 To terminate a job.

Label Operation Accumulator Operand

* 

This indicates that all parameters and data referring to the job introduced by the last preceding *JOB parameter have been read.

21 To terminate a run.

Label Operation Accumulator Operand

****

Any magnetic tapes still open, other than the P.L.T., are closed and the program halts.

Steering Information

The specification of steering information by means of ASTEER and BSTEER parameters has already been described. This subsection is concerned only with the significance of the various steering line items.

In general, the significance of the steering line items for #XPLN is the same as for the PLAN 3 compilers, but as it is possible to specify separately for unconsolidated or consolidated semi-compiled output, the significance of OBJECT is restricted to the former and there is an additional item, CONSOLIDATE. The facilities available are:

| LIST | SHORTLIST | These facilities are the same as with the PLAN 3 compilers, and are discussed in Chapter 7. The resulting printouts are described in Chapter 9. MAP implies CONSOLIDATE, FULLLIST implies MAP and CONSOLIDATE. MAP(OFF) implies CONSOLIDATE. FULLLIST(OFF) implies MAP(OFF) and CONSOLIDATE. |
| MAP | FULLLIST | BINARY implies CONSOLIDATE. |
| MAP(OFF) | |
| FULLLIST(OFF) | |
| BINARY | |

MONITOR

This steering line item also has the same effect as with the PLAN 3 compilers, except that either MONITOR or MONITORX will be called, according to whether the program’s #PMODE directive specifies 15AM or 22AM. As MONITOR is a compact mode routine, care should be taken if #PMODE specifies 15AM that the program is operating in 15-bit address mode and direct branch mode where monitor points occur, and that no attempt is made to monitor areas of store above 32K. MONITORX will operate in either address mode and either branch mode, so similar considerations do not arise when #PMODE specifies 22AM.

OBJECT

The OBJECT steering line item with #XPLN requests the output of unconsolidated semi-compiled segments with segment leaders, thus permitting the creation and maintenance of a file of semi-compiled segments. Output to such a file in response to OBJECT on a steering line will only take place if an output tape specified by an OWFC parameter or an APPC parameter is currently open, and a subfile opened by a WSF parameter is currently open on it. If OBJECT is encountered on a steering line and no tape specified by an OWFC or an APPC parameter is currently open, an error message is given on the line printer, if a line printer has been allocated; otherwise, the error message is given as a console display. If CONSOLIDATE appears without OBJECT on a steering line, output will also be written to the unconsolidated semi-compiled output file under the same conditions as for OBJECT, except that no error message is given if no such file is open.

CONSOLIDATE

The CONSOLIDATE steering line item with #XPLN requests the output of a consolidated semi-compiled program ready for loading. #FINISH is not required in source input as the job terminator parameter indicates that the consolidation phase may be entered. If an output tape specified by an OWFA or an OWFB parameter is currently open, the object program is written to that tape; otherwise a scratch tape will be opened to receive the object program, or, if a scratch tape has already been opened to receive earlier object program output, the object program will be added to that tape.
CONSOLIDATE implies OBJECT, in that unconsolidated semi-compiled segments will be output to the tape specified by the last OWFC or APPC parameter, if such a tape is currently open and has a subfile currently open. Unconsolidated semi-compiled segments are also written to the work tape.

Restrictions

Not more than 32,767 words may be allocated under any one #PROGRAM or #UPPER directive in any one segment.

The total number of words allocated under the #PROGRAM directive in any one program must not exceed 32,767.

Line Printer Error Messages

The following error messages may be output on the line printer in the course of a run:

- **JOBNAME name INVALID**
  The first character of a job name declared in a *JOB parameter is not alphabetic.

- **NO S/C TAPE SPECIFIED**
  An OBJECT steering line has been encountered, and no tape specified by an OWFC or APPC parameter is currently open. Compilation proceeds but no unconsolidated semi-compiled output file is produced.

- **LOWER LIMIT VIOLATION #name**
  The program being consolidated has allocated more than 4096 words of lower store. Consolidation continues, but whether the program can be run successfully or not depends on how the offending lower store is addressed.

- **OVERLAY BLANK CUES #name**
  A segment specified under a #OVERLAY directive is missing from the program being consolidated. The program is abandoned with the message DISPLAY:-- SERV.

- **CONVERGING TABLES #name**
  The combined symbol and branch ahead table is full and cannot be extended. The program being compiled is abandoned and the compiler proceeds to the next job.

- ***TABLE EXCEEDED 1600 WORDS AT LINE xxxx**
  For an explanation of this message see page 20 of Chapter 7.

- **CNSL.TABLE FULL #name**
  The consolidated cue list is full. The program being consolidated is abandoned with the message DISPLAY:-- SERV.

- **MAP/FULLIST TABLE FULL #name**
  The map/fullist table is full. The production of the map and/or fullist is abandoned, and the compiler proceeds to the next job. The object program has already been output.

- **UN.n FULL #name NOT COMPILRED**
  A tape opened for writing has become full. The program’s relative unit number of the deck holding the tape is indicated by n. The job ‘name’ is abandoned and the run is brought to a close; subsequent jobs within the run are not examined. The program ‘#name’ is not present in the output file.

- **GROUP name NOT FOUND**
  The required subroutine group ‘name’ cannot be found on the library tape, MT0. Consolidation continues.

- **ERROR An**
  There is an error in the parameter input. ‘An’ denotes the type of error thus:

  - A1  Parameter not recognized.
  - A2  Parameter operand has incorrect format.
  - A3  RWH parameter does not immediately follow an OWFA or an OWFC parameter, or has a value of X different from that in the immediately preceding OWFA or OWFC parameter; and so is ignored.
  - A4  An end-of-subfile sentinel has been detected on the magnetic tape called by a PLAN or an S/C parameter but the count n specified
in the parameter is not exhausted. (Note: If the subfile contains consolidated semi-compiled program, and all the segments from it are required, this message will appear but should be ignored.)

A5 The accumulator field of a magnetic tape control parameter is not in the range 1 to 6.

A6 An attempt has been made to write a sentinel to the wrong tape; that is, the accumulator field of a WSF or a SFEND parameter specifies a tape that was not opened by an OWFC or an APPC parameter.

The parameter in error is printed in print positions 1 to 64, and is flagged in print positions 65 to 72 with ******** where ????? are four variable characters. The compilation continues if this is possible. If it is not possible to continue with the job, the compiler proceeds to the next job.

There is an error in the parameters or other input such that compilation cannot continue. The present job is abandoned and the compiler proceeds to the next job. Abandonment of a job in this context means that no further input is read from magnetic tape, nor from a paper tape reader or card reader other than the one holding the parameters, and no further compilation takes place; action is still taken on any parameters opening, closing or rewinding magnetic tapes, or opening or closing subfiles. If input intended for the abandoned job is still standing in a paper tape reader or card reader other than the one being used to read parameters, it should be removed if the reader is required also for subsequent jobs.

The type of error which has occurred is indicated by xn, thus:

C1 There is a checksum error in a semi-compiled block on magnetic tape.

C2 There is a checksum error in a semi-compiled block on paper tape or cards.

E2 No file having the file number indicated by X in the parameter is open.

E3 The file indicated for reading is a write file.

E4 There is already a file open with the file number specified by X in an open file parameter.

E5 The file number specified by X in a WSF or SFEND parameter is that of a read file.

E6 The file specified to be opened as a read file is not a composite file.

F1 Two consecutive tape marks have been detected.

F2 There is an unclearable parity error.

F3 A long block has been read.

F4 There is a combination of tape errors.

F5 An unrecognizable sentinel has been read.

F6 An end-of-subfile sentinel is missing.

F7 An end-of-file marker has been encountered.

F8 The data format is recognized.

F9 Semi-compiled segment terminator not found.

P1 Semi-compiled input from paper tape or cards contains more than 18 words.

P2 The requested segment is not present in this file.

P3 The specified subfile does not contain the correct type of data.
If the message is the result of a parameter error the offending parameter is
printed in print positions 1 to 64, and is flagged in print positions 65 to
72 with "??????" where ???? are four variable characters. If the message
indicates a C2 or P1 error the offending input block is printed, but with
its last two words replaced by a flag in the format "??????" where ????
are four variable characters.

Operating Instructions for #XPLN

1 Load the program library tape containing #XPLN; load any other magnetic tapes required as input, output or
work tapes.

2 Input the message:
   F1 #XPLN #NAME

   where #NAME represents the name of the appropriate search program (generally #TAPE).

3 Load the control parameters and any source or semi-compiled segments which are to be input from paper tape
or cards, in the paper tape reader and/or card reader.

4 If the first parameter is to be read from paper tape, activate the program by the message:
   GO #XPLN 20

   If the first parameter is to be read from a card, activate the program by the message:
   GO #XPLN 21

5 The program allocates the appropriate reader and performs the various actions specified by the control
parameters. Whenever a *JOB parameter is read, the following message is output:
   0#XPLN; DISPLAY:- START OF JOB NAME,XXXXXXX

   where NAME and XMMMMMM are as specified in the *JOB parameter. Whenever a job terminator is read,
one of the following messages is output:
   0#XPLN; DISPLAY:- COMP OK AAAAA##NAMEMMMMMM

   if the compilation has been successfully completed; or
   0#XPLN; DISPLAY:- COMP ER AAAAA##NAMEMMMMMM

   if the compilation has been completed, but errors are present; or
   0#XPLN; DISPLAY:- SERV OK AAAAA##NAMEMMMMMM

   if the job has been abandoned, but is error free up to the point of abandonment; or
   0#XPLN; DISPLAY:- SERV ER AAAAA##NAMEMMMMMM

   if the job has been abandoned, and contains errors.

   AAAAAA is the total of the number of lines printed and the number of blocks written to magnetic tape in
   the course of the job.

6 When the file terminator is read, any magnetic tapes (other than the P.L.T.) remaining open are closed, and
the program halts with the console message:

   0#XPLN; HALTED:- END OF BATCH
EXCEPTION CONDITIONS

The following messages may be output on the console typewriter by #XPLN:

DISPLAY:-- BC

The program being consolidated has an unsatisfied cue. If the unsatisfied cue is a name defined in a #OVERLAY directive the program is abandoned, the message OVERLAY BLANK CUES #name (where #name is the name of the abandoned program) is output on the line printer, and the compiler proceeds to the next job. Otherwise, the missing name is given a value of #75777 and consolidation continues. In either case the SERV display (see step 5 of the operating instructions) is output for the program.

If the program contains both overlay blank cues and other unsatisfied cues, then the DISPLAY:-- BC message is output twice.

DISPLAY:-- NEEDS TAPE *nnnnnn

A magnetic tape with a file name as specified in a parameter has been opened, but its tape serial number has been checked and found to be incorrect. The tape is unloaded, and the remaining decks are searched for the correct tape. If the correct tape cannot be found on any deck, Executive outputs a message asking for it to be loaded.

DISPLAY:-- PL

A paper low condition has been detected on the line printer. The line printer is disengaged. The operator should replenish the paper before reallocating the printer to continue the run.

HALTED:-- ERRO80n

The compiler cannot find one of its overlays (n is the overlay unit number). Repeat the run with the P.L.T. on a different deck. If the halt recurs, the library tape should be recreated, taking care to ensure that the compiler is replaced on the tape as an overlay program.

HALTED:-- L/BLK UNIT n

A long block has been detected on a magnetic tape. The compiler program’s relative unit number of the deck containing the faulty tape is given by n. Abandon the run.

HALTED:-- PAR/F UNIT n

There has been a parity failure on a magnetic tape, and the operator has attempted to resume the run by GO #XPLN. The compiler program’s relative unit number of the deck containing the failed tape is given by n. Abandon the run.

HALTED:-- SB

An end-of-tape marker has been detected on one of the magnetic tapes, so the rest of the batch has been abandoned. No SERV display (see step 5 of the operating instructions) is output for the program being compiled at the time of abandonment. The message UN.n FULL #name NOT COMPILED (where n is the compiler program’s relative unit number of the deck containing the full tape, and #name is the name of the program being compiled at the time of abandonment) is output on the line printer. All files are closed and all peripherals released.

The post-mortem procedure for #XPLN, should it go illegal or loop, is to type in:

GO #XPLN 29

In certain circumstances the compiler will automatically post-mortem itself.

The console output message signifying the end of the post-mortem procedures is:

0#XPLN; HALTED:-- END OF PM

The line printer output and the console log should be forwarded through the normal channels for software error.

If it ever becomes necessary to inhibit the compiler’s automatic post-mortem, this may be done by setting switch 14 on before commencing the run. Should the circumstances then arise which, were switch 14 not on, would initiate the automatic post-mortem, the compiler will halt with the console message:

0#XPLN; HALTED:-- PM

In this event, an Executive core print of the compiler’s area of store should be taken and forwarded, together with the console log and printouts of the magnetic tapes used, through the normal channels for software error.
THE PLAN 4 COMPILER #XPLT

#XPLT is a PLAN 4 batch compiler for machines of 32K and above. It accepts PLAN 2, 3 and 4 source and/or semi-compiled segments from paper tape, cards, magnetic tape, exchangeable disc, fixed disc and semi-compiled segments from consolidated semi-compiled programs on magnetic tape. The selection of segments to be compiled is controlled by parameters read from paper tape or cards. Programs written in 15AM or 22AM or DBM or EBM can be compiled. Output is to line printer and disc file. The compiler does not consolidate, its output being in unconsolidated semi-compiled form, but compiler parameters may be used to call in the consolidator as a successor program and to initiate the consolidation run automatically.

The hardware requirements for #XPLT are:

A central processor with not less than 32K words.
One paper tape reader or one card reader.
One line printer.
One or more exchangeable disc files (optional).
One or more fixed disc files (optional).
One or more magnetic tape decks (optional).
One paper tape reader or one card reader (optional).

The use of the peripherals is described in the sections which follow.

#XPLT is a non-overlay program and can be loaded from any medium. The load device must be added to the above list of hardware requirements if it does not already appear there. The priority of #XPLT as supplied is 93.

Besides all the normal facilities of the PLAN 4 language, #XPLT can generate calling sequences to the following compact mode software packages:

MONITOR.
Input/Output Generator.
Magnetic Tape Housekeeping system.
Storage Device Housekeeping systems (15-bit address mode versions).
Overlay system.
Dump and Restart packages.
Input

The overall unit of compilation is a batch of source and/or semi-compiled segments punched on eight-track paper tape or 80-column cards, or a magnetic tape file, an exchangeable disc file or a fixed disc file. Files for input must conform to the ICL 1900 Series organization standards and must be in subfile format. Files produced by #XPLN and #XPLT, disc and magnetic tape COSY systems are suitable for input to #XPLT. PLAN source and/or semi-compiled segments may be input from magnetic tape or disc under the selective control of parameters on paper tape or cards.

If magnetic tapes written by programs other than those already mentioned are to be used as input, the following points should be noted.

All magnetic tapes used for input must be single reel composite files in 1900 Series standard subfile format: the subfiles may be simple (containing either PLAN source or semi-compiled records, but not both), or composite (containing both source and semi-compiled subfiles). Blocks may be of variable length (maximum length 210 words), with the maximum size of 21 words of source program or 18 words of semi-compiled program.

Cards must be punched in the ICL 64-character card code. A line on paper tape may consist of up to 80 characters and must be terminated by a newline.

A source segment on any medium is deemed to be a set of records starting with a #PROGRAM record and ending with a #END record, except that a #STEER record (which #XPLT may be instructed to ignore) may precede the #PROGRAM record.

Amendments in an amendment subfile associated with a source subfile being compiled are included.

Input files may be opened and closed as required, and up to four such files may be open at any one time.

The input peripheral required to read control parameters (described later) may also be used to read PLAN source and/or semi-compiled segments. The required peripheral, a paper tape reader or a card reader allocated as PTO or CRO, is determined initially by the choice of entry point used (see 'Operating Instructions' on page 28), and is released when either all the input data has been read or there is a change of character peripheral input medium. The character peripheral input medium for parameters can be changed, by a SWITCH parameter, at any time after the first two paper tape lines or cards have been read. When a SWITCH parameter is encountered in the input data the current reader is released and one of the other type is allocated. PLAN source and/or semi-compiled segments may be read from the reader that currently holds the parameters, or from a reader of the other type if so specified in a parameter. In the latter case the required reader is allocated when the parameter is read and is released after the number of segments specified by the parameter has been read.

Steering Information

The compiler facilities that may be called by the STEER or SEGSTEER parameter or by the steering record of a source segment are LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or FULLLIST and/or MAP. The LIST, SHORTLIST, OBJECT, FULLLIST and MAP facilities are the same as those for the PLAN 3 compilers and are discussed in Chapter 7 except that the FULLLIST and MAP facilities will produce records suitable for input to #XPSR. The MONITOR steering item also has the same effect as with PLAN 3 compilers, except that either MONITOR or MONITORX will be called, according to whether the program's #PMODE directive specifies 15AM or 22AM. As MONITOR is a compact mode routine, care should be taken if #PMODE specifies 15AM that the program is operating in 15-bit address mode and direct branch mode where monitor points occur, and that no attempt is made to monitor areas of store above 32K. MONITORX will operate in either address mode and either branch mode, so similar considerations do not arise when #PMODE specifies 22AM.

The printouts resulting from the LIST, SHORTLIST and MONITOR facilities are described in Chapter 9.

The facilities to be called are specified when the source file is created or updated, as described in Chapter 11. Segments with blank steering records are not compiled unless a STEER or SEGSTEER parameter is supplied. If semi-compiled segments are to be copied across from the input to the output file, this too is specified when the source file is created or updated, as described in Chapter 11.

Output

A line printer is allocated as LP0 when the compiler is entered, and is released at the end of the run. In addition to providing any listings that may be requested by steering information, the line printer is used for printing error
messages and various informatory messages (see page 29) and a copy of the input parameters just prior to their being obeyed.

Subfiles of unconsolidated semi-compiled object program are output on a disc file by means of the OUT or APP parameters (see 'Control Parameters'). Such disc files must previously have been allocated by the appropriate file allocator.

To obtain consolidated object program in binary form a separate run using an appropriate consolidator is required. Provision is made, by means of the NEXT control parameters, for the compiler to delete itself and call the consolidator into store at the end of the run. If the BIN parameter is specified, the consolidation may then be entered automatically, however only the last program of a batch may use this facility in any one run.

Control Parameters

The compiler parameters to be read from paper tape or cards conform to the same generalized format as the parameters for PLAN 3 compilers discussed on page 21 of Chapter 7. Each parameter line starts at the first character position of a paper tape line or at the first column of a card. Paper tape lines are terminated by a newline character and have a maximum permitted size of 128 internal characters; beta shift and delta shift characters other than $, ] , \rightarrow\rightarrow\rightarrow\rightarrow, newline and horizontal characters are not accepted.

The parameters that are available are as follows:

1. **PROG**

   This parameter is optional. If the PROG parameter is present, it may specify a new object program name and charge number and remains in force until another PROG parameter or (for the last program in a batch) an ENDPJOB parameter is read.

   The parameter has the following format:

   \[
   \text{PROG} \ (\text{NAME},\text{CHARGENO})
   \]

   where \( \text{NAME} = \) the 4-character object program name. The name given in this parameter overrides the program name given in the \#PROGRAM line. If the \text{NAME} given exceeds 4 characters in length the compiler will pick up the first four characters only as the program name. If a \text{NAME} comprising up to 3 characters or no \text{NAME} is given a warning message INVALID 'PROG' will be output.

   \( \text{CHARGENO} = \) the charge number which may be up to 8 alphanumeric characters. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted a charge number of 8 space characters will be assumed.

   If the PROG parameter is absent the compiler will force in a program name of XXXX.

2. **IN, INE, INF or INM**

   This parameter is optional. It defines the file to be opened as the input file. IN and INE define an exchangeable disc file; INF defines a fixed disc file; INM defines a magnetic tape file. The file may contain both PLAN source and semi-compiled segments and remains open until either it is closed by parameter or until the end of the run. Up to four input files may be open simultaneously.

   The parameter has the following format:

   \[
   \text{IN}(X;\text{FILENAME}(<\text{GN}),\text{CSN})
   \]

   or

   \[
   \text{INE}(X;\text{FILENAME}(<\text{GN}),\text{CSN})
   \]

   or

   \[
   \text{INF}(X;\text{FILENAME}(<\text{GN}))
   \]

   or

   \[
   \text{INM}(X;\text{FILENAME}(<\text{GN}),\text{TSN})
   \]
where \( X \) = the file reference number and is used to identify the file in subsequent parameters.

\[ \text{FILENAME} \] = the existing 12-character file name. If the file name is punched with fewer than 12 characters, the compiler will supply spaces at the right-hand end to complete the 12 characters.

\[ \text{FGN} \] = the file generation number, in decimal. For exchangeable disc and fixed disc files this number lies in the range 0 to 4095. For magnetic tape files this number lies in the range 0 to 8388607. If it is omitted, when specifying magnetic tape files, the file generation number will be unchecked. If it is omitted, or specified as \(-1\) when specifying exchangeable disc or fixed disc files, then the file with the highest generation number on-line will be opened. If it is omitted, the associated brackets should also be omitted.

\[ \text{CSN} \] = the cartridge serial number in octal of the file to be opened. It lies in the range 1 to 777777 and may optionally be punched with a preceding \# or *. If it is omitted or specified as zero the serial number will not be checked. If it is omitted the preceding comma should also be omitted.

\[ \text{TSN} \] = the tape serial number, in octal of the file to be opened. It lies in the range 1 to 377777 and may optionally be punched with a preceding \# or *. If it is omitted or specified as zero the serial number will not be checked. If it is omitted the preceding comma should also be omitted.

If a serial number is specified, \#XPLT will check that the file opened is on the correct tape or disc.

OUT, OUTE, OUTF, APP, APPE or APPF

If a program or a batch of programs being compiled is to be listed only, this parameter is not required; if any segment being compiled has OBJECT in its steering line, then this parameter is compulsory. It defines the file to be opened as the output file. Only one output file may be specified for a program. It is possible to request that the output file be closed and another opened in its place. This is done by giving a new PROG parameter for the next program and another OUT, OUTE, OUTF, APP, APPE or APPF parameter which closes any currently open file and subfile and opens the new file. OUT, OUTE, APP and APPE define an exchangeable disc file; OUTF and APPF define a fixed disc. The OUT, OUTE and OUTF parameters cause the new subfile to overwrite the existing contents of the file. APP, APPE and APPF cause the new subfile to be appended to the file leaving the existing contents unchanged. It is not possible to add data to an existing subfile.

The parameter has the following format:

\[
\text{OUT(FILENAME}(\text{FGN1=FGN2}),\text{CSN})
\]
or

\[
\text{OUTE(FILENAME}(\text{FGN1=FGN2}),\text{CSN})
\]
or

\[
\text{OUTF(FILENAME}(\text{FGN1=FGN2})
\]
or

\[
\text{APP(FILENAME}(\text{FGN1=FGN2}),\text{CSN})
\]
or

\[
\text{APPE(FILENAME}(\text{FGN1=FGN2}),\text{CSN})
\]
or

\[
\text{APPF(FILENAME}(\text{FGN1=FGN2})
\]

where \[ \text{FILENAME} \] = the existing 12-character filename of a permanent file which is to be opened for writing. If the filename is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

\[ \text{FGN1} \] = the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, or if it is expressed as \(-1\), then the highest-numbered generation of the specified file on line will be opened.
FGN2 = the new file generation number to be written to the output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding # sign should also be omitted. If it is omitted the file is opened in overlay mode and the generation number remains unchanged.

CSN = the serial number, in octal, in the range octal 0 to octal 777777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked. If a serial number is specified, #XPLT checks that the file opened is on the correct disc.

If FGN1 and FGN2 are both omitted the associated brackets are also omitted; thus, for example:

OUTE(Filename,CSN)

APPF(Filename)

The version number of the file will be set to zero if FGN2 is present, or will be increased by one if FGN2 is omitted.

If the CSN field is punched in an OUTF or APPF parameter it will be treated as a parameter error; a fixed disc unit serial number should therefore not be quoted.

Note: Under GEORGE 2, an OUTE or OUTF parameter may take the following form:

OUTE(Filename(*),CSN)

or

OUTF(Filename(*),CSN)

The compiler will attempt to open the file nominated for the semi-compiled with a generation number of 1. If this fails it will automatically attempt to open the file nominated with a generation number of 2, and so on, up to 4.

4 REN

This parameter is optional. It causes the output file in the parameter immediately preceding this parameter to be renamed. A REN parameter is acceptable only if an OUTE or OUTF parameter is also present.

The parameter has the following format:

REN(Filename(FGN/VN))

where Filename = the new file name which is to overwrite the existing file name on the output file. It may be up to 12 characters long. If it is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted the output file will be given a generation number of zero. FGN must not be expressed as -1.

VN = the version number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the solidus should also be omitted. If it is omitted, the output file will be given a version number of zero.

If FGN is omitted, and VN is present, the solidus must also be present; thus:

REN(Filename(/VN))

If FGN and VN are both omitted the associated brackets are also omitted; thus:

REN(Filename)

If a REN parameter is present, the file generation number and version number to be written to the output file is determined solely by the REN parameter; if, when REN is present, and OUTE or OUTF parameter specifies FGN2, that field of the OUTE or OUTF parameter will be overridden.

5 WSF

This parameter causes a new subfile to be opened on the output file. The output file must already be open when this parameter is given. A subfile remains open until either a WSF parameter opens another subfile or the file is closed. Semi-compiled output to a file occurs only if a subfile is open to receive it.
The parameter has the following format:

\texttt{WSF(SUBFNAME)}

where \texttt{SUBFNAME} = the name of the new subfile to be opened. It may be up to 12 characters long. If it is punched with fewer than 12 characters the compiler will supply space at the right-hand end to complete the 12 characters.

\section*{CLOSE and CAF}

CLOSE causes a specified input file to be closed; CAF causes all files to be closed. These parameters are not likely to be needed very often since an input file is closed automatically when opening another with the same value of \texttt{X} and all files are automatically closed at the end of the job.

The parameter has the following format:

\texttt{CLOSE(X)}

or

\texttt{CAF(X)}

where \texttt{X} = the reference number of the input file to be closed.

\section*{EXTE}

This parameter specifies an exchangeable disc on which any extension of the output file may be made. If this parameter is omitted, then extension will only take place on the disc containing the last logical bucket of the output file.

The parameter has the following format:

\texttt{EXTE(CSN)}

or

\texttt{EXTE(CSN)}

where \texttt{CSN} = the cartridge serial number in octal of the disc on which the extension is to be made. It lies in the range 1 to 777777 and may optionally be punched with a preceding \# or *. If it is specified as zero, then extension will be uncontrolled.

\section*{REW}

This parameter causes the specified magnetic tape file to be rewound and positioned immediately after the header label.

The parameter has the following format:

\texttt{REW(X)}

where \texttt{X} = the reference number of magnetic tape file.

\section*{SWITCH}

This parameter causes the basic peripheral to be changed from paper tape to cards or vice versa. Thus if the parameter occurs on paper tape, subsequent parameters will be read from cards.

The parameter has the following format:

\texttt{SWITCH}

\section*{STEER}

This parameter controls the steering of all segments from the point at which it occurs until another PROG or STEER is encountered or unless it is overridden for specific segments by a SEGSTEER parameter.

The parameter has the following format:

\texttt{STEER(Steering information)}

where \texttt{Steering information} = LIST and/or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST. The effect of the steering information is as described in Chapter 7 except that if MONITOR is specified and \#MODE is 22AM, MONITORX will be called. If it is omitted, then steering will be controlled by the \#STEER directive given in the source input.
SEGSTEER

This parameter controls the steering of those segments specified by the following PLAN, S/C or ALL parameter. After a SEGSTEER parameter has been executed for those segments, steering is controlled by the current STEER parameter.

The parameter has the following format:

SEGSTEER(Steering information)

where

Steering information = LIST and/or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST. The effect of the steering line is as described in Chapter 7 except that if MONITOR is specified and #PMODE is 22AM, MONITORX will be called. If it is omitted, then steering is controlled by the current STEER parameter or the #STEER directive given in the source input.

PLAN

This parameter causes the specified PLAN source segments on a magnetic medium or on cards or paper tape to be compiled.

Where the PLAN source segments are on a magnetic medium, the parameter has the following format:

PLAN(X;SUBFNAME/SEGNAME,N)

where

X = the reference number of the file.

SUBFNAME = the 12-character subfile name of the subfile located on the file. If it is omitted the preceding semi-colon, SEGNAME together with the preceding solidus and N with the preceding comma should also be omitted. This will cause all PLAN source segments in the file to be compiled.

SEGNAME = the 12-character segment name of the first segment to be compiled. If it is omitted then the preceding solidus, N and the preceding comma should also be omitted. This will cause all PLAN source segments in the subfile to be compiled.

N = the number of successive PLAN source segments to be compiled starting at SEGNAME. If SEGNAME is present and N together with its preceding comma is omitted then a value 1 is assumed.

If a composite subfile on a magnetic tape is specified, a simple subfile containing PLAN source is sought and the segments in that single subfile compiled.

Where the PLAN source segments are on paper tape or cards, the parameter has the following format:

PLAN(TR,N)

or

PLAN(CR,N)

where

TR denotes the source segments are on paper tape.

CR denotes the source segments are on cards.

N = the number of successive PLAN source segments that are to be compiled. If it is omitted together with its preceding comma then a value of 1 is assumed.

If the segments are on the same medium as the PLAN parameter, they should immediately follow it in the input stream.

Each source segment must start with #PROGRAM, which may be preceded by #STEER, and end with #END. If OBJECT is requested, the semi-compiled output is to the subfile that is currently open.

S/C

This parameter causes the specified semi-compiled segments on a magnetic medium or on paper tape or cards to be copied to the output file.

Where the semi-compiled segments are on a magnetic medium, the parameter has the following format:

S/C(X;SUBFNAME/SEGNAME,N)
where \( X \) = the reference number of the file.

\[
\text{SUBFNAME} = \text{the 12-character subfile name of the subfile located on the file. If it is omitted the preceding semi-colon, SEGNAME together with the preceding solidus and } N \text{ with the preceding comma should also be omitted. This will cause all semi-compiled segments in the file to be copied to the output file.}
\]

\[
\text{SEGNAME} = \text{the 12 character segment name of the first segment to be copied. If it is omitted then the preceding solidus, } N \text{ and the preceding comma should also be omitted. This will cause all semi-compiled segments in the subfile to be copied to the output file.}
\]

\[
N = \text{the number of successive semi-compiled segments to be copied starting at SEGNAME. If SEGNAME is present and } N \text{ together with its preceding comma is omitted, then a value of 1 is assumed.}
\]

Where the semi-compiled segments are on paper tape or cards, the parameter has the following format:

\[
S/C(TR,N)
\]

or

\[
S/C(CR,N)
\]

where \( TR \) denotes the semi-compiled segments are on paper tape.

\( CR \) denotes the semi-compiled segments are on cards.

\[
N = \text{the number of successive semi-compiled segments that are to be copied to the output file. If it is omitted together with its preceding comma then a value of 1 is assumed.}
\]

14 **ALL**

This parameter causes the specified PLAN source segments to be compiled and semi-compiled segments to be copied to the output file.

The parameter has the following format:

\[
\text{ALL(X,SUBFNAME/SEGNAME,N)}
\]

where \( X \) = the reference number of the file.

\[
\text{SUBFNAME} = \text{the 12-character subfile name of the subfile located on the file. If it is omitted the preceding semi-colon, SEGNAME together with the preceding solidus and } N \text{ with the preceding comma should also be omitted. This will cause all PLAN source segments to be compiled and semi-compiled segments to be copied to the output file.}
\]

\[
\text{SEGNAME} = \text{the 12-character segment name of the first segment to be compiled or copied. If it is omitted then the preceding solidus, } N \text{ and the preceding comma should also be omitted. This will cause all PLAN source segments to be compiled and semi-compiled segments to be copied to the output file.}
\]

\[
N = \text{the number of successive segments to be copied starting at SEGNAME. If SEGNAME is present and } N \text{ together with the preceding comma is omitted then a value of 1 is assumed.}
\]

15 **AMPL**

This parameter allows amendments to be made to a source segment held on disc or magnetic tape without altering the input file.

The parameter has the following format:

\[
\text{AMPL(X,SUBFNAME/SEGNAME,N)}
\]

where \( X \) = the reference number of the file to be amended.

\[
\text{SUBFNAME} = \text{the 12-character subfile name of the subfile to be amended. If it is omitted the preceding semi-colon, SEGNAME together with the preceding solidus, and } N \text{ with the preceding comma should also be omitted. This will cause the specified amendments to be made to the first source segment and also cause all PLAN source segments in the file to be compiled.}
\]

M3 4322(7.72) Chapter 8 23
SEGNAME = the 12-character segment name of the segment to which the amendments are to be made. If it is omitted then the preceding solidus, N and the preceding comma should also be omitted. This will cause the specified amendments to be made to the first source segment in the subfile and also cause all other PLAN source segments in the subfile to be compiled.

N = the number of successive segments to be compiled starting at SEGNAME. If SEGNAME is present and N together with the preceding comma is omitted then a value of 1 is assumed.

The parameter is followed immediately by the amendments for the specified segment. Each line to be amended is specified by a #ALTER line. Source lines to be deleted or to be preceded by insertions are referenced by the line number. Amendment lines must be given in ascending order for a segment.

Where lines are to be deleted, the #ALTER line has the following format:

#ALTER A,N

or

#ALTER A-B

where

A = the line number of the first line to be deleted.

N = the number of successive lines to be deleted.

B = the line number of the last line to be deleted.

Any #ALTER line may be followed by PLAN source lines which are to replace the deleted lines. All lines following the #ALTER line will be inserted, until the next #ALTER or until a line #FINISH is found signalling the end of the amendments for this segment.

Where lines are to be inserted, the ALTER line has the following format:

#ALTER A

where

A = the line number of the line in front of which insertions are to be made.

The #ALTER line will be followed by PLAN source lines to be inserted before line A.

If the AMPL parameter is used and a listing is required, inserted lines are given a line number of !!!!!; deleted lines retain the line number, but also have asterisks in print positions 6 and 7. Each block of omitted lines is preceded and followed by two blank lines.

EXAMPLE
AMPL(1:SUBA/SEG,D,3)
#ALTER 1,1
#PROGRAM PROG/SEG,D(22AM,EBM)
#FINISH
AMPL(1:SUBA/SEG,G)
#ALTER 17

DATE(2)

LPCON 2/0,0,121,0/AREA,3

#ALTER 25,2

#ALTER 52-69

EXIT 1 0
#FINISH

This will cause the following to occur:

The #PROGRAM line in SEG,D in subfile SUBA on file reference number 1 is to be replaced by a new #PROGRAM line, and three segments, starting at SEG,D are to be compiled. SEGG in the same subfile requires two additional lines to be inserted before line 17. Two lines, 25 and 26, from the same segment are to be deleted. Lines 52 to 69 are to be deleted, and the EXIT line is to replace the deleted lines.
**NEXT**

This parameter is optional. It determines whether or not the compiler is to delete itself and call in a successor program at the end of the run.

If the NEXT parameter is absent, the compiler halts at the end of the run.

If the NEXT parameter is present, then at the end of the run the compiler deletes itself with a FIND message, to initiate the loading of a successor program. The compiler program provides the 'FI' portion of this message; the variable portion is obtained from the NEXT parameter. The NEXT parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests or charge numbers may be included in the parameter, in addition to the fields discussed below.

The generalized format of the NEXT parameter is:

```
NEXT(Variable portion of DELTY 'FIND' message)
```

This may usually be formalized as:

```
NEXT(#ABCD#NAME N)
```

where

- `#ABCD` = the name of the program to be brought into core when the compiler deletes itself. This will usually be `#XPCK`, the consolidator program.
- `#NAME` = the last four character of the name of the file containing the successor program. The name of this file must be of the form PROGRAM NAME. `#NAME` may be omitted where NAME and ABCD are the same.
- `N` = the amount of core store into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program's request block.

The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

The references made in a NEXT parameter will be cancelled if a new PROG parameter has been encountered later.

**BIN, BINE, BINF, ABN, ABNE or ABNF**

This parameter may be used only when each of an OUT, OUTE or OUTF parameter and a NEXT parameter are present, and the NEXT parameter specifies a suitable consolidator (for example, `#XPCK`); it is optional in those circumstances. It causes the consolidator program, when it is loaded, to be entered without operator intervention, and so results in automatic consolidation. If automatic consolidation is specified, the compiler's last output subfile will become the first inclusive mode subfile presented to the consolidator. If the OUT, OUTE or OUTF and the NEXT parameters are present with this parameter, but the NEXT parameter specifies a program other than a suitable consolidator, then this parameter is accepted but is ineffective.

The parameter has the following format:

(a) `BINE(FILENAME(FGN))`

or

(b) `BINF(FILENAME(FGN))`

or

(c) `BINE`

or

(d) `BINF`

If form (a) is used, the consolidated program will be output in binary to the specified exchangeable disc file.

If form (b) is used, the consolidated program will be output in binary to the specified fixed disc file.

If form (c) is used the consolidator's workfile will be opened on exchangeable disc and the consolidated program will be held in store ready to run.
If form (d) is used the consolidator's workfile will be opened on fixed disc and the consolidated program will be held in store ready to run.

If BIN is used instead of BINE or BINF it will be treated as BINE.

If the first three characters of the parameter are ABN instead of BIN, the previous contents of the file will be preserved and a new subfile added at the end of the file.

In the above parameter formats:

FILENAME = the existing 12-character file name of a permanent file which is to be opened and overwritten by the consolidator's output. If the file name is punched with fewer than 12 characters the compiler will supply spaces at the right-hand end to complete the 12 characters. The file must be on the same cartridge or unit as that specified by the OUT, OUTE or OUTF parameter.

FGN = the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted, in which case the associated brackets should also be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on the cartridge or unit containing the compiler's output file will be opened.

The version number of the file specified by a BIN, BINE or BINF parameter will be increased by one.

INC, INCE, INCF, LIB, LIBE or LIBF

These parameters define the input disc subfiles for automatic consolidation: INCE or INCF gives a subfile in inclusive mode, LIBE or LIBF gives a subfile in exclusive mode, that is, a library subfile. If INC is used it will be treated as INCE; if LIB is used it is treated as LIBE.

The parameter has the following format:

INCE(FILENAME(FGN1).SUBFNAME)

or

INCF(FILENAME(FGN1).SUBFNAME)

or

LIBE(FILENAME(FGN1).SUBFNAME)

or

LIBF(FILENAME(FGN1).SUBFNAME)

where

FILENAME = the 12 character file name of the file on which the subfile is held.

FGN1 = the file generation number currently existing on the file. It may be omitted in which case the associated brackets should also be omitted.

SUBFNAME = the 12 character subfile name of the subfile to be consolidated.

The order of consolidation will be

(a) The last output subfile produced by the compilation, in inclusive mode.

(b) The subfiles defined in the INCE/F and LIBE/F parameters, in the appropriate mode, in the order in which the parameters appear.

If no LIBE/F parameter is given, a subfile named SUBROUTINES in a file SUBGROUPS-RS will be assumed unless a #COMPLETE is present in the source input to the compiler. The presence of a #COMPLETE in the input to the compiler will override any LIBE/F parameters presented.

Exclusive mode requests consolidation of only those segments having a force-in bit set or for which there is an unsatisfied cue, that is, as from a library. Inclusive mode requests consolidation of all segments within the subfile.

ENDPROG

This parameter causes all files that are still opened to be closed, the peripherals to be released and the end of run procedure for #XPLT to be executed (that is, either halt or chain in a subsequent program). This parameter may occur once only, as the last parameter for a batch or for a program.
The parameter has the following format:

```
ENDPROG
```

**COMMAND ISSUING**

When operating under GEORGE 3, #XPLT will, on reading a file specifying parameter (that is, any of INF, INF,
INM, OUTE, APPE, OUTF and APPF), issue a suitable ASSIGN or ONLINE command for the file specified. An
ASSIGN will be issued if a serial number is absent; an ONLINE if the serial number is present. For example,
```
INF(2,SOURCEFILE) will produce
AS *ED5,SOURCEFILE(0)
```
```
OUTE(SEMICOMPILED(6),123456) will produce
OL *ED2(WRITE),(123456,SEMICOMPILED(6))
```

The files for which ASSIGN or ONLINE commands are issued are limited to files specified as general local names.
Absolute names (for example, :FRED.SEMICOMP) or GEORGE workfiles (!) will cause #XPLT to halt PE
(parameter error).

The command issuing facility may be suppressed by setting switch 11 of word 30 on before entering #XPLT.

**AUTOMATIC CONSOLIDATION**

In order to obtain automatic consolidation the following conditions must be met:

1. A NEXT parameter specifying the consolidator must be given.
2. A BINE, BINF, ABNE or ABNF parameter must have been given.
3. At least one output subfile must have been opened.
4. LIBE, LIBF, INCE, INCF parameters, if present, must be the last parameters presented to #XPLT except for
   the ENDPROG. The steering information for the consolidator is placed in the semi-compiled output file,
   so this file should not be closed by parameter if automatic consolidation is requested.

**BATCH COMPILATION**

Each program is introduced by a PROG parameter specifying the new object program name and charge number, and
terminated by the next PROG parameter or (for the last program of the batch) an ENDPROG parameter. The
ENDPROG must occur only once, as the last parameter.

The effect of a PROG parameter other than the first is as follows:

1. Terminate compilation of previous program with the usual DISPLAY - COMP/SERV ... message.
2. Close any currently open output subfile (but not the file).
3. Cancel any current STEER or SEGSTEER parameter.
4. Cancel references to previous NEXT, BINE, BINF, INCE, INCF, LIBE and LIBF parameters. These
   parameters will have been printed but are otherwise ignored, so automatic consolidation can only be
   obtained for the last program of the batch and all parameters relating to it must come after the last PROG.

It is also possible to request that the output file be closed and another opened in its place. This is done merely by
giving another OUTE, OUTF, APPE or APPF parameter, which closes any currently open output file and subfile,
and opens the new file.

**Restrictions**

Not more than 32,767 words may be allocated under any one #PROGRAM or #UPPER directive in any one
segment.

The total number of words allocated under the #PROGRAM directive in any one program must not exceed 32,767.
Operating Instructions for #XPLT

1. Load the parameters on the appropriate reader.
2. Load any tape or disc files required for the run.
3. Load #XPLT into store. #XPLT is a non overlay program and can be loaded from any medium.
4. If it is required to suppress the command issuing facility (see above), input:
   
   ON #XPLT 11

5. If the first parameter is on paper tape, activate the compiler by the message:
   
   GO #XPLT 20

6. If the first parameter is on cards, activate the compiler by the message:
   
   GO #XPLT 21

When the parameters have been entered and the PROG parameter has been read the following message is output on the console typewriter:

0#XPLT; DISPLAY:- START JOB NAME, CHARGE NO

where NAME and CHARGENO are those given by the PROG parameter.

7. At the end of compilation of each program, one of the following messages is output by the compiler:

   0#XPLT; DISPLAY:- COMP OK AAAAAAA #NAME CHARGENO
   
   if compilation has been successfully completed; or

   0#XPLT; DISPLAY:- COMP ER AAAAAAA #NAME CHARGENO
   
   if compilation has been completed but errors are present; or

   0#XPLT; DISPLAY:- SERV OK AAAAAAA #NAME CHARGENO
   
   or

   0#XPLT; DISPLAY:- SERV ER AAAAAAA #NAME CHARGENO
   
   if compilation has been abandoned.

Note: AAAAAAA is the total number of lines printed and blocks written during the run.

8. At the end of the run the line printer is released and one of the following messages is output on the console typewriter:

   0#XPLT; HALTED:- OK
   
   if automatic consolidation is not requested; or

   0#XPLT; DELETED:- FI MESSAGE
   
   if a NEXT parameter is given.

In addition the following line printer messages may be output during compilation:

INVALID 'PROG'

The PROG parameter has a program name which does not conform to the convention of starting with a letter and consisting of alphanumeric characters only or has a program name of up to 3 characters in length or has no program name. The compiler will force in a default name of XXXX, and compilation continues.

FILENAME *CSN USED AS u

Shows details of file just opened where u = unit number.

FILE u EXTENDED TO s BLOCKS

Shows details of extension of output file.

STEERING BKT. b

Steering information for the consolidator starts in bucket b of the output file.
EXCEPTION CONDITIONS

The following messages may be output on the console typewriter:

DISPLAY:- MT *TSN  
Tape of right name but wrong serial number opened. Mount the tape requested  
if it is not already on-line. #XPLT unloads the original tape.

HALTED:- ED *CSN  
or  
:- FD *CSN  
A file of the requested name has been found on a device of the wrong serial  
number. Make the right device available and GO #XPLT.

HALTED:- HH  
The run has reached its normal end but the automatic consolidation requested  
has been suppressed due to an S1, S2 or S3 error (see line printer messages  
below).

HALTED:- MT  
Requested magnetic tape not on line. Make the right tape available and  
GO #XPLT.

HALTED:- PE  
HALTED:- XX  
HALTED:- ZZ  
These are accompanied by a line printer message.

The following messages may be output on the line printer:

#ALTER AT FAULT  
An amendment was not presented in ascending line number order or a line  
has an incorrect format.

ALTERS NOT ALL DONE  
An attempt has been made to alter a non-existent line. The compilation  
continues.

CONVERGING TABLES #name  
The combined symbol and branch ahead table is full and cannot be extended.  
The program being compiled is abandoned and is accompanied by  
HALTED:- ZZ.

DISC TRANSFER ERRn  
FILE u  
 Accompanied by HALTED:- XX on the console typewriter.  
n is the reply character as described in the Direct Access Manual.

ERROR xy **aaaa**  
General error message where xy is a two-character code as follows:

xy     Reason               Compiler action
A1     Incorrect parameter  HALTED:- PE
A2     Incorrect steering parameter  HALTED:- PE
A3     Reader not specified as CR  
or TR  
A4     More segments have been  
requested than are available  
in the input subfile  
Those segments found before the  
end of subfile are compiled, and the  
program continues. This is not  
necessarily an error.
B1     AMPL not followed by a  
#ALTER or a #FINISH LINE  
HALTED:- XX
C1     S/C record of more than 18 words  
found  
Offending record printed, then  
HALTED:- ZZ
C2     Checksum error  
As for C1
C3     Faulty slow semi-compiled input  
(no title record)  
As for C1
C4     Disc source record format error  
As for C1
C5     Disc input format error  
As for C1
F1     No space SFD on APP/APPE/  
APPF file  
HALTED:- ZZ
F2     Reference made to an input file  
not open  
HALTED:- ZZ
xy  Reason  Compiler action
F3  WSF given when no output file open  HALTED:-- ZZ
M1  Tape marks found unexpectedly on input tape  HALTED:-- XX
M2  Invalid sentinel on input tape  HALTED:-- XX
M3  End of file sentinel found unexpectedly on input tape  HALTED:-- XX
M4  Faulty data format (semi-compiled magnetic tape input)  HALTED:-- XX
S1  Segment not found on input tape
S2  Subfile not found on input tape
S3  Subfile or segment not found on input disc file
T1  Backspace failure on input tape  HALTED:-- XX

FILE u NOT EXTENDED
ERR n  Accompanied by HALTED:-- XX on the console typewriter
FILENAME NOT OPENED
ERR n  Accompanied by HALTED:-- ED or FD on console typewriter.
*NO OBJECT AS NO OUTPUT SUBFILE
No output subfile has been opened so no semi-compiled program will be written to disc.

RENAME FAILED  Accompanied by HALTED:-- ED or FD on console typewriter.

*TABLE EXCEEDS 1600
WORDS AT LINE XXXX
For an explanation of this message see page 1 of this Chapter.
TAPE READ ERR n  Accompanied by HALTED:-- XX on the console typewriter.
FILE u

n is the reply character as described in the Direct Access manual.

To retry, input GO #XPLT

PLAN 3 PROGRAMS AND THE PLAN 4 COMPILERS

Programs written in PLAN 3 can be compiled by the PLAN 4 compilers. As means of specifying the initial address mode and the checks to be performed on each segment's address mode are not available in PLAN 3, a steering segment is not required for this purpose in PLAN 3, although one may be present for other purposes, for example the specification of overlays. The PLAN 4 compilers will therefore accept PLAN 3 programs with or without a steering segment, and will compile such programs to operate initially in 15-bit address mode and direct branch mode.

COUNTER-MODIFIERS

Programs written in PLAN 4 may be required to work in compact mode or in extended data mode. This choice of address modes affects the process of modification.

In compact mode, a 15-bit modifier is used. Advantage may be taken of the 9-bit/15-bit division of a word as a counter-modifier to control the process of modification by BUX or BDX instructions, and of the 2-bit/7-bit/15-bit division of a word as a character counter-modifier to control character modification by BCHX instructions. In extended data mode, a 22-bit modifier is used; bits 0 and 1 of the word are still used as the character address in character modification. Thus there is no room for a count to be contained in the same word, and the BUX, BDX and BCHX instructions operate differently (see Chapter 4).

As stated earlier in this chapter, PLAN 4 provides additional macro-instructions to facilitate the handling of two-word counter-modifiers in extended data mode. The PLAN 4 macro-instructions BUX, BDX and BCHX achieve similar effects in extended data mode programs to those obtained by the user of BUX, BDX and BCHX instructions in compact mode programs; the LDCM macro-instruction enables the loading of a count in one accumulator and a 22-bit modifier in another accumulator to be programmed in a single statement, in a manner compatible with the use of the BUX, BDX and BCHX macro-instructions. These macro-instructions are described in Chapter 4.
As PLAN 4 programs may be required to operate in either address mode, it is necessary to provide facilities for the compilation both of 15-bit modifiers, with provision for a count in the same word, and of 22-bit modifiers. The discussion in the next paragraph is necessary in order to explain how the compilation of modifiers, as described in the following paragraphs, is controlled. Similar considerations apply to the compilation of character counter-modifiers.

The modifier portion of an index word as written on a PLAN coding sheet may be either an absolute expression or a relative expression (see page 14 of Chapter 2). In the latter case, the expression is evaluated by the compiler into two parts, an absolute part and a relative part. The relative part is the relativizer, discussed in Chapter 7 and in the description of compiler listing in Chapter 9. The absolute part consists of a number indicating the position allocated to the symbolic identifier within the area of store associated with the relativizer, adjusted by any signed decimal or octal integer which may have formed part of the relative expression. For example, if the index word 

`10/OUTPUT+3` is being evaluated, and OUTPUT has already been allocated as the twenty-first word of lower variable data for the segment (word 20 LV), then the compiler will evaluate the relative expression `OUTPUT+3` as an absolute part, 23, and a relativizer, LV.

When an index word is encountered in a constant data statement or in a literal in PLAN 4 the following rules apply.

1  Index word expressed as /MODIFIER or 0/MODIFIER (no count specified, or zero count).

(a) First #PROGRAM line of the segment specifies `(22AM)` or `(15AM,22AM).

   The index word is regarded as requiring a 22-bit modifier.

   (i) MODIFIER an absolute expression. If MODIFIER evaluates to more than 22 bits it is truncated to 22 bits and a 'J' error is flagged.

   (ii) MODIFIER a relative expression. If after evaluation the absolute part as discussed above is more than 22 bits, it is truncated to 22 bits and a 'J' error is flagged.

(b) First #PROGRAM line of the segment specifies `(15AM)`, or fails to specify an address mode.

   The index word is regarded as requiring a 15-bit modifier.

   (i) MODIFIER an absolute expression. If MODIFIER evaluates to more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

   (ii) MODIFIER a relative expression. If after evaluation the absolute part as discussed above is more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

2  Index word expressed as COUNT/MODIFIER.

   The index word is regarded as requiring a 9-bit count and a 15-bit modifier, irrespective of the declared address mode of the segment.

(a) First #PROGRAM line of the segment specifies `(15AM)` or fails to specify an address mode.

   (i) MODIFIER an absolute expression. If MODIFIER evaluates to more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

   (ii) MODIFIER a relative expression. If after evaluation the absolute part as discussed above is more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

(b) First #PROGRAM line of the segment specifies `(22AM)` or `(15AM,22AM)`.

   (i) MODIFIER an absolute expression. If MODIFIER evaluates to more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

   (ii) MODIFIER a relative expression. If after evaluation the absolute part as discussed above is more than 15 bits it is truncated to 15 bits and a 'J' error is flagged.

   If the relativizer is other than LV, R2:LV, LP or R2:LP an 'H' error is flagged. This is a warning that the modifier may evaluate to more than 15 bits; if it is known that the final evaluation will be to 15 bits or less, the H error flag may be ignored.

In cases 1 (a) (ii), 1 (b) (ii), and 2 (a) (ii) above, the absence of a J error flag should not be taken as a categorical indication that the modifier has compiled successfully: it is still possible that the absolute part plus the relativizer may evaluate to more than the permitted number of bits.

Similar considerations to the foregoing apply to the compilation of character counter-modifiers.
BRANCH MODES

For an explanation of branch modes, see Chapter 1, pages 7 and 8.

When a PLAN 4 program is compiled to operate in extended branch mode, the following rules are observed:

1. Branches between segments are compiled as replaced branches.
2. Branches within a segment are compiled as relative branches, except that:
   (a) If the difference between the branch destination address and the address of the location holding the branch instruction is more than 8191, then the branch is compiled as a replaced branch.
   (b) If the location holding the branch instruction is not within the program operation store, then the branch is compiled as a replaced branch. Thus all branch instructions held in the upper or lower preset data store are compiled as replaced branches.
   (b) If a branch instruction is specified for compilation as a replaced branch, then it is so compiled.

A branch instruction may be specified for compilation as a replaced branch by inserting a colon (:) as the first character of the operand field, preceding the branch destination address.

It is possible to write segments which may be consolidated into both extended branch mode and direct branch mode programs. It is not until the consolidation stage that the type of branch (direct, relative or replaced) of each branch instruction is finally determined. The semi-compiled output from the compiler contains the information as to whether a branch, if the segment is consolidated into an extended branch mode program, is to be relative or replaced; this information will either be acted upon or ignored by the consolidator, depending on the branch mode of the program into which the segment is being consolidated. The presence of a colon in the operand field of a branch instruction statement does therefore not inhibit the semi-compiled form of the segment containing it from inclusion in a direct branch mode program.

The compiler #XPLN, not implementing extended branch mode features, does not accept a colon in the operand field of a branch instruction.

Replacers

A replacer is a location which contains in its least significant 22 bits an address which may be used as the destination address of a replaced branch instruction. Replacers are generated automatically by the compiler and the consolidator, and stored below location 16,384 of the program under a special relativizer, RR. If a segment specified as acceptable for consolidation in both extended branch mode and direct branch mode programs is included in a direct branch mode program, then its semi-compiled form will contain provision for replacers, but the replacers will be excluded from the direct branch mode object program.

A replacer is generated for each cued location. Cued locations comprise:

1. The first location of a segment’s program operation store (this is normally the first or only location generated by the statement immediately following the second #PROGRAM directive of a segment). The cue name in this case is the segment name.

2. The first or only location generated by the statement immediately following a #CUE directive.

In order to minimise additional coding where branches to CUENAME + N are required, PLAN 4 provides expanded forms of the #PROGRAM and #CUE directives which generate replacers for a special number of locations consecutive with the cued location; see the descriptions of these directives in Chapter 6. For example, if a segment contained the directive statement:

#PROGRAM

N

where N represents an absolute expression, then a branch to SEGMENTNAME + N would use the Nth replacer after that for the segment name. If N is allowed to be too big, however, and no branches to the intervening locations are required, it could cause the generation of an excessive number of unwanted replacer storage locations: in such circumstances the use of a #CUE directive before the statement representing the required branch destination is preferable.
Chapter 9 Compiling and Testing Programs

INTRODUCTION

This chapter is concerned with some of the preliminary steps involved in preparing a program before operational running.

The first section deals with some of the rules for punching PLAN source programs into paper tape or cards. This is followed by descriptions of the listings produced when PLAN programs are compiled.

The MONITOR and tracing facilities that are available as program testing aids are dealt with in further sections. The chapter concludes with a description of the library programs that can be used when amending source programs on paper tape or cards.

PUNCHING THE SOURCE PROGRAM

Source programs may be punched into paper tape or 80-column cards. The standards that should be adopted when using these media are described in separate sections below.

Paper Tape

The following standards are essential:

1. One newline character must be punched at the end of each program statement (further newlines may be punched if a tape operated typewriter is to be used for hard copy, and line spacing is required).
2. Where punching errors are detected at the time of punching, the output tape should be moved back, the incorrect tape characters obliterated by delete characters, and this should be followed by the correct punching.
3. Each length of tape should be terminated by the standard end-of-tape record which is (on eight-track paper tape):

```
Direction of reading
TC4
Five Deletes
Newline
```

4. Paper tape is punched according to the appropriate character set code (see Part 4, Appendix 1).
5. The horizontal tab punching facility should be used to skip over unused fields or parts of fields, though this may also be accomplished by punching the appropriate number of spaces. When the compiler reads source program on paper tape, it recognizes the horizontal tab characters and converts each line to card image form, spacing it out in a 20-word area. The recognized horizontal tab positions are as follows:

- Column 7: beginning of operation field
- Column 13: beginning of accumulator field
- Column 16: beginning of operand field
- Column 36: beginning of comment
- Column 61: beginning of further comment
Take as an example this section of program written in PLAN:

```plaintext
#PROGRAM
LDX 1 COST
SBX 1 MAY
BNZ 1 PD4
DB3 2 LDCT 2 INDEX(3)
CALCULATEPRICE
```

The punching for this would be:

`#PROGRAM`-
`LDX 1 COST`-
`CALCULATEPRICE`-
`SBX 1 MAY`-
`BNZ 1 PD4`-
`DB3 2 LDCT 2 INDEX(3)`-

where

- `\|` signifies newline
- `->` horizontal tab
- `\|` space

**NOTE:** Where an entry completely fills a field a tab skips to the next field but one. For example:

```plaintext
ENTRY
```

is punched `ENTRY -> 4`

not `ENTRY -> 4`

The following standards are optional:

6. A run out of about nine to twelve inches should lead each tape.

7. Gaps may be inserted between statements by punching RUN OUT or PUNCHED FEED or the equivalent key. This action causes feed holes to be punched, and is useful for subsequent identification of the beginning of statements for correction purposes.

8. Vertical tab and form feed characters may be inserted at any point between statements, instead of the newline character, in order to improve the appearance of a listing produced on the tape editing equipment. The codes produced on paper tape for these characters are subsequently treated in the same way as a newline character by the compiler.

**80-column Cards**

1. Cards are punched in the ICL 1900 Series 64-character set code (see Part 4, Appendix 1). Blank columns are read into the computer as spaces.

2. The card format is given on the PLAN coding sheet (see page 4 of Chapter 2).

3. One card is punched for each line of information.

**COMPILER LISTINGS**

All PLAN compilers can provide a listing of the source program together with an interpretation of its semi-compiled form. There are two basic forms of this listing, one produced by PLAN 4, PLAN 3 and PLAN 2 compilers, and the other by PLAN 1 compilers. PLAN 4, PLAN 3 and PLAN 2 compilers may alternatively provide an abbreviated form of listing, known as a short list. When the listing or the short listing is requested in combination with any facility which results in consolidated object program output, PLAN 4 compilers and PLAN 3 compilers with integral consolidators also print a consolidated program summary; if a PLAN 3 or PLAN 2 compiler without an integral consolidator is used, a similar summary may be printed during the consolidator run.

PLAN 4 compilers and certain PLAN 3 compilers can alternatively or additionally provide a listing of the source program together with a listing of the resulting object program after consolidation. This is known as a full list.
An intermediate facility available with PLAN 4 compilers and with some PLAN 3 compilers is the store map. This is post-consolidation analysis of the storage requirements of each segment, showing the address relative to program datum allocated to each symbolic name used.

A store map is included in a full list, and may be called alone or in combination with a list or short list. The method of requesting the various facilities, and the PLAN 3 compilers with which the store map and full list facilities are available, are given in Chapter 7 in the section entitled 'Steering Lines'. The different printouts are described separately below.

The PLAN 4, PLAN 3 and PLAN 2 Compiler Listing

During the compilation of programs written in PLAN 4, PLAN 3 or PLAN 2 a listing of the whole source input as punched, together with an interpretation of its semi-compiled form, can be printed if specified in the steering line. An example of such a compiler listing is included in Appendix 5.

The contents of the PLAN 4, PLAN 3 and PLAN 2 compiler listing are as follows:

1. With compilers which have basic peripheral input, the first line is a copy of the steering line. If program and segment steering lines are both present preceding the first segment, both are printed. Compilers with input from magnetic tape or cassette tape have their steering information shown in the source file editor program printout, but not reproduced in the compiler listing. Compilers with input from disc print their steering information as the first line of print unless they are under the control of external steering obtained from a SUB parameter (see page 39 of Chapter 7), in which case the printing of that parameter will have already indicated the steering information. (With PLAN 4 compilers the factors controlling the printing of steering information are somewhat more complex than this, but the information is printed either as a steering line or in a parameter.)

2. Following the steering line(s) or as the first line of print is a heading line which shows the time and date of compilation, and identifies the compiler used. If the central processor used has no internal clock, the time is printed as zero.

3. Subsequent lines list the program. On the left-hand side is printed a straight transcription of the punching of the source program (including comments), and on the right-hand side is printed an interpretation of the semi-compiled form of constant data and program instructions.

The details printed and their positions on the printout are as follows:

<table>
<thead>
<tr>
<th>Print Positions</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>The number of the line in the source program segment, starting at 1.</td>
</tr>
<tr>
<td>9 to 88</td>
<td>Transcription of the source program punching:</td>
</tr>
<tr>
<td></td>
<td>(a) Characters 1 to 80 for card input.</td>
</tr>
<tr>
<td></td>
<td>(b) Up to character position 72 for paper tape input.</td>
</tr>
<tr>
<td>73 to 87 (PLAN 2) or 89 to 103 (PLAN 3)</td>
<td>A heading indicating the common area that is in use, printed on a separate line when there is a change of common area.</td>
</tr>
<tr>
<td>93 to 96</td>
<td>If the line is in error, a letter that indicates the type of error (see the Table on page 6 for a list of error codes). Up to three types of error can be flagged on any one line. If the compiler detects more than three types of error on a line it prints an asterisk in place of the fourth alphabetical error code.</td>
</tr>
<tr>
<td>98 to 102</td>
<td>The decimal address of the line within a major directive area, starting at 0. If the statements are interspersed, the sequence of addresses is carried forward, e.g.</td>
</tr>
<tr>
<td></td>
<td>#LOWER 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>#PROGRAM 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>\LOWER 2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>105 to 116</td>
<td>The details printed are as follows, according to the type of statement:</td>
</tr>
<tr>
<td></td>
<td>(a) Constant data: the contents of the word in octal.</td>
</tr>
</tbody>
</table>
(b) Instructions:
an interpretation of the semi-compiled form,
  FFF X M AAAA for non-branch instructions,
  FFF X AAAAAA for branch instructions.

(c) Macro-instructions: all the instructions generated are printed.

(d) Packages: the calling sequence generated is printed.

A two-character code as follows:

(a) LV, LP, UV, or UP (Lower Variable, Lower Preset, Upper Variable, or Upper Preset respectively):
    Operand refers to an address in the specified area.

(b) LT: Operand points to a location in which a literal is stored.

(c) PR (or R1): Branch to a previously specified label.

(d) BA: Branch to an unspecified label (Branch Ahead).

(e) R2: Reference to the common area named in the nearest previous heading.

(f) R3: Branch to library software.

The PLAN 4 and PLAN 3 compilers throw to a new page after each sixtieth line of print, unless
switch 18 is set, in which case they throw to a new page on detecting a punching in channel 8 of the
printer control loop. (Thus non-standard stationery may be used for compilation lists.) The PLAN
2 and PLAN 1 compilers throw to a new page after each fifty-seventh line of print.

4 Following the listing of each segment, after the #END directive has been printed, appears a summary
of the segment's storage requirements. PLAN 4 and PLAN 3 compilers first throw to a new page,
unless the MAP facility has been requested. This summary appears under the following headings,
which are printed underlined:

(a) 0 AB
    All symbolic identifiers given absolute values under #DEFINE and #SET directives are listed,
    together with the decimal representation of their values. The letter S is printed to the left of
    identifiers that appeared in a #SET directive. For #SET identifiers only the final value is printed.

(b) n LV
    where n is the number of words allocated to lower variables. All the lower variable symbolic
    names are listed, together with their start addresses within the area. The letter I is printed
    to the left of implicitly defined lower variables.

(c) 0 R2:LV
    All lower common variables are listed in the form:
    /n name
    where n indicates the total number of words allocated to the common area and name denotes
    its block name.
    Under each common area are given the symbolic locations within the area, together with their
    relative starting positions.

(d) n LP
    where n is the number of words allocated to lower preset data. The list of symbolic names is
    given, as for LV.

(e) n LT
    where n is the number of words allocated to literals.
    On the right-hand side of the page, and not directly underneath the heading, are listed all the
    literals, represented in octal. Each literal is accompanied by its relative position within the
    literal area, and if it references a symbolic location within another area, this is indicated
    by LV, LP, R2, etc. If there are more than 75 literals in the segment, the first 75 (or each
    multiple of 75) will have been printed already as an interruption in the segment's listing, and
the literal table will have been zeroized for re-use; only the contents of the literal table accumulated since it was last zeroized are printed here.

(i) 0 R2:LP
All lower common constants are listed, in a similar format to R2:LV.

(g) n PR or n R1
where n is the number of words allocated for the program instructions.
All labels used are listed under this heading, together with their relative positions within the program area. The labels are in the order in which they are encountered, whether in the label field or operand field.

(h) n UP
where n is the number of words allocated to upper preset data. The list of symbolic names is given, as for LV.

(i) 0 R2:UP
All upper common constants are listed, in a similar format to R2:LV.

(j) n UV
where n is the number of words allocated to upper variable data. The list of symbolic names is given, as for LV.

(k) 0 R2:UV
All upper common variables are listed, in a similar format to R2:LV.

(l) 0 BA
All branch ahead labels that are unsatisfied within the segment are listed, including branches to library software.

Listings produced by the PLAN 4 compilers have the following additional summary headings:

(m) 0 R2:EV
This is placed between the R2:UV and the BA summaries. All upper common variables defined under the #ELASTIC directive are listed, in a similar format to R2:LV. The same area may appear under R2:UV in other segment summaries if in those segments it is defined under #UPPER instead of #ELASTIC.

(n) n RR
This appears only if the segment is specified on its first #PROGRAM line as being intended to work correctly if consolidated into a program whose overall branch mode is EBM. It is placed after the BA summary. The number n is the number of replacers generated by branch instructions (whether the branch instructions be in the program operation store or in the upper or lower preset data store); it excludes replacers generated by #CUE and #PROGRAM directive statements.

Note: With those compilers with which it is available, if the MAP facility is being used, then only item (e) of the above summary is printed. Expanded details of the other items are given instead in the store map, following the consolidated program summary.

The last item in the listing is the error line. This consists of the letter 'E' followed by 24 letters or spaces. The purpose of this line is to give an at-a-glance indication of whether the segment is error-free or not, and the types of errors that are present (naturally this refers only to PLAN syntax errors, not to errors of logic).

The 24 types of errors are denoted by the letters A to X (see the Table below), which also appear beside the lines in error in the program listing. If a letter appears in the listing, whether once or more, it is also printed in the appropriate position in the error line. If there is no error of a particular type in the program, then a space appears in the appropriate position in the error line.

The PLAN 2 compiler prints a seven in the appropriate position for each type of error that was present and a zero for each type not present.

Where compiler errors are flagged in the error line, it is usually necessary to amend the source program and then recompile.

With PLAN 4 and PLAN 3 compilers a 'TABLE EXCEEDED' message may also appear on the error line, to the right of the error code indications. For an explanation of this message please
6 With PLAN 4 compilers the error line may be followed by either or both of the following error messages:

   SEGMENT BRANCH MODE INCORRECT
   SEGMENT DATA MODE INCORRECT

   For an explanation of these messages see under the #PMODE directive in Chapter 6.

   If further segments are present, these are listed in turn as in 1 to 6 above. PLAN 4 and PLAN 3 compilers throw to a new page to commence the listing of a new segment.

   PLAN 3 compilers with output on disc additionally produce a program summary error line when the #FINISH directive is encountered (#XPLF and #XPLM) or when the last source subfile for the program has been read (#XPLX and #XPLZ). The format of the line is the same as that produced by PLAN 4 and PLAN 3 compilers with integral consolidators as part of the consolidated program summary, and is described on page 9.

   Error Codes used in PLAN Compiler Listing

<table>
<thead>
<tr>
<th>Code</th>
<th>Position in error line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>23</td>
<td>Expression value is indeterminate (e.g. undefined symbol).</td>
</tr>
<tr>
<td>B</td>
<td>22</td>
<td>Expression type is indeterminate (e.g. combination of symbols of different types).</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>Expression type is incorrect for this particular line.</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>Expression has incorrect form (e.g. illegal character).</td>
</tr>
<tr>
<td>E</td>
<td>19</td>
<td>Expression value is too large.</td>
</tr>
<tr>
<td>F</td>
<td>18</td>
<td>Single implicit-defined location used as double length.</td>
</tr>
<tr>
<td>G</td>
<td>17</td>
<td>Operation code incorrect.</td>
</tr>
<tr>
<td>H</td>
<td>16</td>
<td>Operand may be too big for this instruction.</td>
</tr>
<tr>
<td>I</td>
<td>15</td>
<td>X-field incorrect for this instruction.</td>
</tr>
<tr>
<td>J</td>
<td>14</td>
<td>Operand is too big for this instruction.</td>
</tr>
<tr>
<td>K</td>
<td>13</td>
<td>Modifier field is incorrect.</td>
</tr>
<tr>
<td>L</td>
<td>12</td>
<td>Modifier is not allowed with this instruction.</td>
</tr>
<tr>
<td>M</td>
<td>11</td>
<td>Symbol has too many characters or an illegal character.</td>
</tr>
<tr>
<td>N</td>
<td>10</td>
<td>Symbol has been defined more than once.</td>
</tr>
<tr>
<td>O</td>
<td>9</td>
<td>Statement is not PLAN format or is out of position.</td>
</tr>
<tr>
<td>P</td>
<td>8</td>
<td>If from paper tape or a macro-instruction, illegal character or line too long. If from cards, card sequence or identity error.</td>
</tr>
<tr>
<td>Q</td>
<td>7</td>
<td>Cue name defined more than once.</td>
</tr>
<tr>
<td>R</td>
<td>6</td>
<td>Segment or program terminator missing, or END statement missing in I/O Generator sequence.</td>
</tr>
<tr>
<td>S</td>
<td>5</td>
<td>I/O Generator statement in error or macro-instruction has wrong number of parameters.</td>
</tr>
<tr>
<td>T</td>
<td>4</td>
<td>I/O Generator editing error.</td>
</tr>
<tr>
<td>U</td>
<td>3</td>
<td>Macro-instruction has been defined more than once.</td>
</tr>
<tr>
<td>V</td>
<td>2</td>
<td>Macro-instruction table is full.</td>
</tr>
<tr>
<td>W</td>
<td>1</td>
<td>Sterling constant has error.</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>Format error in pseudo-operation statement.</td>
</tr>
</tbody>
</table>

   The Short PLAN 4, PLAN 3 and PLAN 2 Compiler Listing

   If SHORTLIST appears in the steering line, then the PLAN 4, PLAN 3 and PLAN 2 compilers produce a listing in the format already described, but the list is abbreviated so that only the following items appear:
1. The steering lines, if the compiler has basic peripheral input.
2. The heading line.
3. All lines commencing with a # character.
4. All lines flagged for error.
5. With the PLAN 2 compiler only, the lines indicating the common area in use.
6. The summaries of the segments' storage requirements.
7. The error line for each segment.

PLAN 3 compilers with output on disc print the program summary error line after the short listing of the last segment.

It should be noted that if both LIST and SHORTLIST are specified in the steering line, only the short compiler listing is printed out.

**PLAN 4, PLAN 3 and PLAN 2 Compiler Listing on a 96 Print Position Printer**

A PLAN 4, PLAN 3 or PLAN 2 compiler listing or short listing may be output on a 96 print position printer if switch 21 is set. Provided that LIST or SHORTLIST appears on the steering line, a listing or shortlisting of the program being compiled is then produced in the format already described, except that the semi-compiled interpretation is transferred from print positions 98 to 119 into print positions 58 to 79, with a colon as a separator in print position 57 and a space in print position 80. Characters from character positions 49 to 72 of the source line will therefore not be printed, but compilation is unaffected.

**The Consolidated Program Summary**

If a compiler listing or short listing is specified in combination with a facility which results in consolidated object program output, PLAN 4 and PLAN 3 compilers with integral consolidators print out a summary of the consolidated program after the listing of the last segment, having first thrown a new sheet. If MAP or FULLLIST are specified in the steering information, compilers which implement these facilities will print out the consolidated program summary before printing the store map or the full listing.

The consolidated program summary comprises:

1. The name of the object program.
2. The core store requirement.
3. The list of relativizers.
4. The consolidated cue list.
5. The program summary error line.

These components of the consolidated program summary are described separately below.

For programs compiled by compilers without integral consolidators (the PLAN 4 compiler #XPLT and the compilers listed in the table on page 58 of Chapter 7) a similar printout, differing in detail, may be obtained from the consolidator program run. For details, see the documentation of the appropriate consolidator.

**THE CORE STORE REQUIREMENT**

The core store requirement of the program is given on a separate line. For a non-overlay program which did not have BINARY in its steering information, this line is in the format:

```
CORE XXXXXX
```

For an overlay program, or for a non-overlay program which had BINARY in its steering information, the line is in the format:

```
CORE YYYYYY INTERMEDIATE XXXXXX
```

The number XXXXXX indicates the number of words of core store required to load the program in G.P.L. load form. The number YYYYYY indicates the number of words required to load the program in binary form.
THE LIST OF RELATIVIZERS

The list of relativizers is printed in a single line across the page. The relativizers are used to determine the final positions and sizes of program areas at run-time. Areas that consist of segments and common areas associated with cue names are given in the consolidated cue list. The other areas of the program are compiled under ten headings that are given in the list of relativizers, in the following order from left to right (Note: this order cannot be guaranteed in future compilers; it is also different in some overlay programs: see Appendix 8):

<table>
<thead>
<tr>
<th>Relativizer Description</th>
<th>Non-overlay</th>
<th>Segment lying within overlay area</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW (Lower Working Area, used when FORTRAN and PLAN segments are combined)</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>LV (Lower Variable Area)</td>
<td>02</td>
<td>06</td>
</tr>
<tr>
<td>R3 or SP (Spare - not in use at present)</td>
<td>03</td>
<td>07</td>
</tr>
<tr>
<td>LP (Lower Preset Area)</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>LT (Literals Area)</td>
<td>23</td>
<td>27</td>
</tr>
<tr>
<td>R6 or SP (Spare - not in use at present)</td>
<td>33</td>
<td>37</td>
</tr>
<tr>
<td>UP (Upper Preset Area)</td>
<td>00</td>
<td>05</td>
</tr>
<tr>
<td>R8 or SP (Spare - not in use at present)</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>UV (Upper Variable Area)</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>R# or SP (Spare - not in use at present)</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

Each of the above relativizers is followed in the list by a decimal number that specifies the relevant area's start location relative to the program's datum. So long as R3, R6, R8 and R# are not in use, the numbers by their sides specify the start locations of, respectively, the Lower Common Variable Area, the Lower Common Preset Area, the Upper Common Preset Area and the Upper Common Variable Area.

The number of words in each area is not indicated, but it may be determined, in conjunction with the cue list, by subtracting the appropriate start positions.

THE CONSOLIDATED CUE LIST

The consolidated cue list is made up of a number of lines, each consisting of an octal number, a decimal number and a symbolic name.

The octal number contains eight digits. The first two digits represent the cue type, as follows:-

<table>
<thead>
<tr>
<th>Cue Type Description</th>
<th>Octal</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program cues (i.e. segment titles)</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>Entry cues (i.e. nCUE lines)</td>
<td>05</td>
<td>5</td>
</tr>
<tr>
<td>Common areas in the Upper Variable Area</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>Common areas in the Upper Preset Area</td>
<td>07</td>
<td>7</td>
</tr>
<tr>
<td>Common areas in the Lower Variable Area</td>
<td>08</td>
<td>8</td>
</tr>
<tr>
<td>Common areas in the Lower Preset Area</td>
<td>09</td>
<td>9</td>
</tr>
<tr>
<td>Blank cues (e.g. missing segments)</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The next six octal digits represent either the start address of the appropriate segment or common area, or the address of a CUEd instruction.

The decimal number is the decimal equivalent of the last six octal digits.

The symbolic name is either the name of a segment or common area, or a cue name.

It may be noted that, except with disc compilers, the value allocated as the start address for blank cues is #75777, so that if an attempt is made to enter a missing segment or subroutine at run time, the program goes illegal. With disc compilers, the value allocated is the address of a word containing a SUSWT 2HQQ instruction, which is inserted in the object program; so that if an attempt is made to enter a missing segment or subroutine at run time, the program halts with the message 'HALTED QQ'. If in a PLAN 4 program the "error segment" facility (see under nERRORSEG and nPROGRAM in Chapter 6) is used, then the "error segment" appears among the program cues, and there are no blank cues.
THE PROGRAM SUMMARY ERROR LINE

The program summary error line is similar in format to the segment error line described on page 5 of this chapter, except that the initial character E is replaced by the words PROGRAM ERRORS. Any error code which has appeared on the error line of any segment of the program being consolidated is repeated in the program summary error line. If the program is completely free of PLAN syntax errors the line appears as:

PROGRAM ERRORS NONE

The PLAN 4 and PLAN 3 Store Map

If MAP is specified in the steering information for a compiler in which the facility is implemented, a printout is produced as follows:

1 The consolidated program summary.

2 A summary of the program's request slip. This appears on two lines: the first line shows the program name, account number if applicable, priority and total core requirement, and the second line shows the total number of each type of basic peripheral which has been requested under PERIPHERAL directives. (Note: this print does not show the program's relative numbers for the peripherals requested, nor does it include peripherals allocated dynamically.)

3 For each segment in turn:
   (a) A heading line showing the program and segment names.
   (b) A summary of the segment's storage requirements in a format similar to that produced by the compiler listing, described on page 4 of this chapter, but expanded to include the following information:
      The start address of each area of store and the address allocated to each symbolic name used, relative to the program datum, is shown in octal and in decimal.
      In the analysis of the PR area, those entries which refer to cue names are preceded by a C.
      The literal table is not printed in the store map.

If the MAP facility is specified for some segments of the program being compiled but not for others, then for those segments which are not being mapped a line is printed in the following format:

PROGRAM #name / segmentname UNMAPPED

A similar line is printed for all library subroutines called.

If after compilation a segment is rejected by the consolidator, then a line in the following format is printed:

SEGMENT segmentname NOT ACCEPTED

An example of a store map may be found in Appendix 5, at the end of the illustration of a full listing.

The Full PLAN 4 and PLAN 3 Compiler Listing

If FULLIST is specified in the steering information for a compiler in which the facility is implemented, a printout is produced as follows:

1 The consolidated program summary.

2 A summary of the program's request slip as described for the store map.

3 For each segment in turn, a listing somewhat similar to that produced for a PLAN 4, PLAN 3 or PLAN 2 compiler listing, but with the following differences:
   (a) Instead of an interpretation of the semi-compiled form of program instructions and preset data, all program instructions and preset data are shown in their compiled and consolidated form, i.e. with all addresses relative to the program datum. Where a preset data item is an index word, the octal representation is supplemented by a decimal representation of the address portion.
   (b) The run-time address, relative to the program datum, of each program instruction and preset data item is shown in decimal representation.
(c) The segment's storage summary is produced as described for store map, and in addition, the literal table is mapped. The literal table is printed out in its octal representation, together with the decimal address, relative to the program datum, of each literal. If the literal is an index word it is represented in its consolidated form, that is the address portion is relative to the program datum; the decimal equivalent of the address portion is also shown.

(d) Following the segment's error line, a second error line is printed in the following format:

```
SEGMENT ERRORS 2 A 1 C 2 E
```

where the number preceding each error code gives the number of times that errors of that type have been flagged in the full listing of the segment. If the segment is free of PLAN syntax errors the line appears as:

```
SEGMENT ERRORS NONE
```

If the program is an overlay program, then the segments being full listed will appear in ascending order of unit number within area number, with the segments within each overlay unit in the sequence in which they were declared under the &OVERLAY directive and the permanent segments last, irrespective of the order in which they were presented to the compiler.

If the FULLLIST facility is specified for some segments of the program being compiled but not for others, then for those segments which are neither being full listed nor mapped a line is printed in the following format:

```
PROGRAM #name / segmentname UNMAPPEd
```

A similar line is printed for all library subroutines called.

If after compilation a segment is rejected by the consolidator, then a line in the following format is printed:

```
SEGMENT segmentname NOT ACCEPTED
```

4 A line showing, for each class of error that has been flagged in any of the full listed segments of the program, the number of times that errors of that class have been flagged. The line has the following format:

```
ERRORS IN #name 4 A 2 B 1 C 3 E
```

Note that the totals are accumulated from only those segments which have been full listed. If no PLAN syntax errors have occurred in all the full listed segments of the program, then the line is printed as:

```
ERRORS IN #name NONE
```

An illustration of a full compiler listing may be found in Appendix 5.

The PLAN 1 Compiler Listing

The PLAN 1 compiler listing (of which there is an example in Part 4, Appendix 5) is made up of lines of up to 96 characters. Each complete listing of a segment consists of:

1 A heading line giving the date, and the input and output peripherals used in compilation.
2 Listings of source statements and an interpretation of the semi-compiled form of these, in parallel.
3 A compilation summary for the segment. The last line of the summary is always:

```
END OF COMPILATION.
```

In detail the listing format is as follows:

1 THE HEADING LINE

This consists of (from left to right):

(a) The words:

```
PLAN 1 (compiler name #m) LISTING
```

where m is a character identifying a particular version of the PLAN 1 compiler being used.

(b) Abbreviations specifying the input and output peripherals used in compilation, thus : CR/CP.

The following standard abbreviations are used:

```
Input:
Card Reader CR
Tape Reader TR
```

Chapter 9
Output: Card Punch CP
       Tape Punch TP

If 'no object output' has been specified, two hyphens (--) replace the output peripheral abbreviation.

(c) The date of compilation (as given by Executive).

An example of a heading line is:

PLAN 1 (XPLP #1) LISTING TR/CP. 01/08/66

2 LISTING OF STATEMENTS

At least one line appears in the listing for each source statement compiled. Completely blank cards and paper tape lines which, when expanded to card image form, are found to consist entirely of space characters are ignored for purposes of compilation and not listed. Certain types of statements (Variable Data statements containing more than one item and Preset Data statements generating more than one word of data) take more than one line for listing the compiler response.

The general format of a line of statement listing is as follows (from left to right):

(a) 24-character semi-compiled statement field divided into:

   (i) A 6-character error field. (This is blank if the compiler has found no format errors in the statement.)

   (ii) An 18-character field giving an interpretation of the semi-compiled statement. (This is blank in the case of directive statements only.)

(b) 72-character source statement field. (This will be blank only if the source statement has already been printed once and further lines are necessary to give the interpretation of the semi-compiled statement.) If characters 65 to 72 of the source statement are space characters, these positions are used to print the statement count; otherwise the statement count is omitted.

This format may be schematically represented as follows:

<table>
<thead>
<tr>
<th>96-character line</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-character semi-compiled statement</td>
</tr>
<tr>
<td>6-character Error Field</td>
</tr>
</tbody>
</table>

The detailed usage of these fields is as follows:

(a) Semi-compiled Statement Field

(i) Error Field

Six code letters are used to flag different types of error that the compiler may detect in the source statement. Each is printed in a particular character position in the listing error field when it occurs, and if a given error type is found anywhere in the segment, the appropriate error code letter also appears in the error summary line (see below).

The error codes are:

<table>
<thead>
<tr>
<th>Letter code</th>
<th>Print position in error field (left to right)</th>
<th>Type of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>1</td>
<td>Label Field Error</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>Function Field Error</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>Accumulator Field Error</td>
</tr>
<tr>
<td>O</td>
<td>4</td>
<td>Operand Field Error</td>
</tr>
<tr>
<td>M</td>
<td>5</td>
<td>Modifier Field Error</td>
</tr>
<tr>
<td>R</td>
<td>6</td>
<td>Read Error</td>
</tr>
</tbody>
</table>

Chapter 9

11
Error types L, I, A, M and R may only appear on lines giving the source statement. Where a source statement gives rise to several lines of listing, code O is used to indicate an error in that item in the source operand field to which the computer response on the flagged line refers. The exact significance of error types L, I, A, O and M depends on the type of statement being compiled (see the summary in the Table on page 9). Note that some types of error cannot occur with some types of statement.

The Read Error has the same significance for all types of statement but is used in a different way according to whether source is being read from paper tape or cards, as follows:
<table>
<thead>
<tr>
<th>Type of Statement</th>
<th>Reason for Error</th>
<th>Label Field</th>
<th>Function Field</th>
<th>Accumulator Field</th>
<th>Operand Field</th>
<th>Modifier Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Statement of Segment</td>
<td>Not a PROGRAM Directive</td>
<td>Error only possible if this is an instruction</td>
<td>Error only possible if this is an instruction</td>
<td>Not a PROGRAM and/or illegal program or segment name format</td>
<td>Error only possible if this is an instruction</td>
<td></td>
</tr>
<tr>
<td>Compiler Action</td>
<td>XCO taken as program and segment name</td>
<td>(see below)</td>
<td>(see below)</td>
<td>XCO taken as program and/or segment name</td>
<td>(see below)</td>
<td></td>
</tr>
<tr>
<td>Directive</td>
<td>Not recognized directive</td>
<td>None</td>
<td>None</td>
<td>Various types of format error depending on type of directive</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Compiler Action</td>
<td>Takes no action on statement</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Clear Storage Allocation</td>
<td>First character of label field not space</td>
<td>None</td>
<td>First character of accumulator field not space</td>
<td>Incorrect name, incorrect allocation expression, or incorrect terminating character</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Compiler Action</td>
<td>Label (if of correct format) will be defined with value = address of next preset word</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Preset Data</td>
<td>Incorrect label, e.g. &gt; 5 characters, or first character not alphabetical</td>
<td>None</td>
<td>First character of accumulator field not space</td>
<td>Incorrect format of item</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Compiler Action</td>
<td>Label will be ignored</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Incorrect label, e.g. &gt; 5 characters or first character not alphabetical</td>
<td>Function mnemonic not recognized</td>
<td>Accumulator field entry incorrect for this instruction type</td>
<td>Operand expression error or incorrect for this instruction type</td>
<td>Non-branch instructions only, Operand field terminated by incorrect character, modifier entry not allowed or format incorrect</td>
<td></td>
</tr>
<tr>
<td>Compiler Action</td>
<td>Label will be ignored</td>
<td>Instruction compiled as though NULL (123)</td>
<td>Will ignore erroneous entry and make X = 0</td>
<td>Will generally give zero operand value</td>
<td>Modifier entry ignored</td>
<td></td>
</tr>
</tbody>
</table>

Summary of PLAN 1 Field Errors
Reading from Paper Tape

Either the statement as read in character mode would take more than 128 characters in the store (this is generally the result of an omitted newline character),
or the statement as adapted to card image form (in accordance with horizontal tabs, etc.) takes more than 72 characters,
or the statement contains an illegal character, e.g. a $ shift character which is neither horizontal tab, nor one of the recognized stops, nor one of the characters $][

Reading from Cards

Either Program Identity Check Failure (the program identity of this statement is not the same as that of the last statement read),
or Sequence Check Failure (the sequence number of this statement is less than that of the last statement read). These checks are not made only if the relevant fields on the cards are not punched.

(ii) Field for interpretation of semi-compiled statement

Usage depends on the type of statement:

Directive Statement
Always blank.

Variable Storage Allocations Statement

A new line is used for each item of Storage Allocations in the statement (the items being considered from left to right), as follows:

<table>
<thead>
<tr>
<th>Character Position</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>Octal relative address of this symbol = number of words allocated to lower variables before this.</td>
</tr>
<tr>
<td>5 to 10</td>
<td>Spaces</td>
</tr>
<tr>
<td>11 to 14</td>
<td>Number of words allocated (in octal).</td>
</tr>
<tr>
<td>15 to 18</td>
<td>Spaces</td>
</tr>
</tbody>
</table>

Preset Storage Statement

A new line is used for each successive word of preset storage generated. The field is used as follows:

<table>
<thead>
<tr>
<th>Character Position</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>Octal relative address of this word = number of words of preset generated before this.</td>
</tr>
<tr>
<td>5 to 6</td>
<td>Spaces</td>
</tr>
<tr>
<td>7 to 14</td>
<td>Octal representation of compiled value of word.</td>
</tr>
<tr>
<td>15 to 16</td>
<td>Spaces</td>
</tr>
<tr>
<td>17</td>
<td>Load type character.</td>
</tr>
<tr>
<td>18</td>
<td>Space</td>
</tr>
</tbody>
</table>
The load type character in position 17 is A, V, or P, and refers to the type of relativizer that will be added to B12 to B23 of the word when loaded. An index word is the only kind of data item that may be one of these three types.

The load character is determined by the type of expression (absolute, variable or preset) appearing to the right of the solidus in the source item. All forms of data item other than index words are absolute, i.e. the compiled result given in the listing is the form in which the word will be loaded, no relativizer being added. The load type character for these is always A.

Non-Branch Instruction
One line only per source statement, with the format as follows:

<table>
<thead>
<tr>
<th>Character Position</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>Octal relative address of this word = number of words of preset generated so far</td>
</tr>
<tr>
<td>5</td>
<td>Space</td>
</tr>
<tr>
<td>6</td>
<td>Accumulator address (Octal representation of B0 to B2)</td>
</tr>
<tr>
<td>7</td>
<td>Space</td>
</tr>
<tr>
<td>8 to 10</td>
<td>'Natural' octal function code (and modifier, if any) (Representation of B3 to B11)</td>
</tr>
<tr>
<td>11</td>
<td>Space</td>
</tr>
<tr>
<td>12 to 15</td>
<td>Operand (Octal representation of B12 to B23)</td>
</tr>
<tr>
<td>16</td>
<td>Space</td>
</tr>
<tr>
<td>17</td>
<td>Load type character</td>
</tr>
<tr>
<td>18</td>
<td>Space</td>
</tr>
</tbody>
</table>

The load type character (A, V, or P) has the same significance as with data items (explained above). It is determined by the type of operand expression appearing in the source statement.

Branch Instruction
One line only per source statement. The format is only slightly different from that for a non-branch instruction, the function being represented in two octal characters and the operand in five:

<table>
<thead>
<tr>
<th>Character Position</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>Octal relative address of this word = number of words of preset generated before this</td>
</tr>
<tr>
<td>5</td>
<td>Space</td>
</tr>
<tr>
<td>6</td>
<td>Accumulator address (Octal representation of B0 to B2)</td>
</tr>
<tr>
<td>7</td>
<td>Space</td>
</tr>
<tr>
<td>8 to 9</td>
<td>'Natural' octal function code (Representation of B3 to B8)</td>
</tr>
<tr>
<td>10</td>
<td>Space</td>
</tr>
<tr>
<td>11 to 15</td>
<td>Operand (Octal representation of B9 to B23)</td>
</tr>
<tr>
<td>16</td>
<td>Space</td>
</tr>
<tr>
<td>17</td>
<td>Load type character</td>
</tr>
<tr>
<td>18</td>
<td>Space</td>
</tr>
</tbody>
</table>

* Natural octal representation is derived by considering B3 to B11 (which includes the modifier) as three octal digits. For example, LDCH with 1 as a modifier would be printed as 121 in this field, i.e.

```
  B3
001 010 001
```

In the list of function codes LDCH is given as 024 (modifier not considered).
(b) Source Statement Field

This field is a 72-character printout of the source statement as read by the compiler. In the case of source read from cards, this corresponds to the contents of columns 1 to 72. A source statement read from paper tape is first adapted to 72-character format, taking account of horizontal tabs.

If, and only if, characters 65 to 72 of the statement are all spaces, a statement count is printed in this position in the listing. The statement count has the format

****nnnn

where nnnn is a four-digit decimal number equal to the number of non-blank statements read so far, including this one.

Since only a small proportion of statements run out beyond column 64, the statement count is printed against most lines of statement listing and it is easy to see what its value should be in those cases where it is omitted. However, it should be borne in mind that, for instance, the regular use of the horizontal tab point at column 61 as a starting point for short comments might result in large tracts of the listing being devoid of statement counts.

3 THE-compilation SUMMARY

The compilation summary consists of the following items:

(a) The line

VARIABLE AREA: nnnn (OCTAL)

where nnnn represents a four-digit octal number equal to the number of words of clear storage allocated in the segment compiled. The total of preset storage allocated may be obtained by adding one to the relative address of the last word of preset (this will generally be in the line immediately preceding #END or #COMPLETE).

(b) Symbol List

All symbols defined in the segment (which have not subsequently been ousted; see #OUST directive, Chapter 6) are listed here in alphabetical order with their octal values and the code A, V or P indicating the type of each (absolute, variable or preset respectively). If the same symbol has been defined more than once in the segment (whether to the same or different values) a warning letter, M, appears against it in the symbol list.

The listing format for each symbol is

M P nnnn SYMBOL

where

M only appears in case of multidefinition.
P stands for A, V, or P as explained above.

nnnn represents a four-digit octal value. If the value > 4095 (this is permissible for absolute symbols used in data statements) a fifth octal digit will appear.

SYMBOL represents the one- to five-character alphanumeric symbol.

(c) The line

NO ' #ENTRY 0' GIVEN

appears if, and only if, no #ENTRY directive specifying entry point 0 has appeared in the segment.

(d) The heading

UNSATISFIED BRANCHES

appears whether there are any unsatisfied branches or not. If there have been any unsatisfied branches ahead, they are listed one below the other after this heading, each with the following format:

nnnn SYMBOL
where \( nnnn \) is a four-digit octal number, the relative address of the branch ahead instruction, and \textsc{symbol} is the symbolic name (of one to five characters) that introduces the branch ahead expression in the instruction and is not defined in the segment (or is ousted prior to the occurrence of this instruction). It should be noted that unsatisfied \textsc{call} expressions generate extra-segmental cues and are not listed here.

(e) Error summary line, which has the following appearance:

\[
\text{ERROR - LIAOMR}
\]

where each of the characters \texttt{L, I, A, O, M} and \texttt{N} (see page 12) only appears if an error of that type has been detected somewhere in compilation of the segment. If there have been no errors, 'ERRORS-' is printed on its own.

(f) The heading

\texttt{CUES -}

This is followed by a list of all the unevaluated symbols appearing in the operand fields of \textsc{call} instructions and all symbols appearing in the operand fields of \#\textsc{cue} directive statements, together with the octal type and value coding for each as output in the cue block. The format of each line of the list is:

\texttt{tttnnn SymbolicName}

where \texttt{ttt} is a type coding of three octal digits, thus:

\texttt{020} for \textsc{cue} defined symbols

\texttt{000} for \textsc{call} symbols.

\texttt{nnnn} is the cue value.

For \textsc{cues} this is the relative address of the next word of preset following the \#\textsc{cue} directive. For \textsc{calls} this is always \texttt{0000}.

'SymbolicName' is the appropriate alphanumeric symbolic name (of one to eleven characters).

The heading 'CUES -' still appears even if no cues are listed.

(g) Every complete listing of a segment is terminated by the line: END OF COMPILATION.

**THE MONITOR FACILITIES**

The contents of selected areas of core store may be printed out at chosen points during the execution of a program by means of the \textsc{monitor} facilities. This can provide information valuable for the diagnosis of program errors.

\textsc{Monitor} is used mainly at various points in the program to check on the state and contents of data areas, for instance to check that tables used by the program are set up correctly.

\textsc{Monitor} must be specified in the program's steering line so that the \textsc{monitor} subroutine is incorporated in the program during consolidation. If this is not done, \#\textsc{monitor} directives and their associated parameters are ignored. It is thus possible to recompile a proved program, omitting the \textsc{monitor} facilities, without removing the directives and parameters from the source program; the same facilities can then be reinstated, if required, without delay or difficulty.

The core store utilization of the \textsc{monitor} subroutine is 336 words. It comprises:

- 11 words of lower variable store.
- 76 words of lower preset store.
- 15 words of literals store.
- 236 words of program operation store.

In addition, one word of lower variable store is allocated for each segment in which calls to the \textsc{monitor} subroutine are compiled.

There is a full description of the \#\textsc{monitor} directive and its use in Chapter 6 (page 24 and 25).
MONITOR is a compact mode package. Similar facilities for extended data mode and extended branch mode programs are provided in PLAN 4 by MONITORX. Full details of this routine are given in the PLAN Program Development Aids manual.

The MONITOR Printout

The format of the MONITOR printout is as follows:

1 The printout is headed by

     MONITOR PRINT m

     where m is the three-digit or four-digit identifier specified by the user in the #MONITOR directive, and indicates the point in the program occasioning the output.

2 The contents of the accumulators, X0 to X7, are printed. These are always output without being specifically requested, but may be suppressed (see Chapter 6, page 24). The details printed are as for 3 below.

3 The contents of the areas specified in the #MONITOR directive list are printed as follows for each word:

   (a) Location address within the program as a decimal number.
   (b) The contents in instruction form:

     FFF  X  M/AAAA for non-branch instructions,
     FFF  X  AAAAAA for branch instructions.
   (c) The contents as four characters.
   (d) The contents as eight octal digits.
   (e) The contents as a signed, zero-suppressed, decimal integer.
   (f) The contents as a signed, decimal fraction.

The paper is advanced two lines before and after each MONITOR printout. There is a blank line between one area and the next in the printout.

Line printer unit 0 is allocated by MONITOR for the printout. If this is not available, the program halts and types out the message

     MONITOR REQUIRES PRINTER

The operator should then make the line printer available to the program, if possible; otherwise, the switches (see below) should be turned off. In any case, the program should be continued at the next instruction.

The line printer, once allocated, is not released by MONITOR.

The MONITOR Switches

Bits 0 to 9 of word 30 of each program can be used as switches to indicate monitoring requirements. When the MONITOR package is entered, it first tests the switch corresponding to the first digit of the identifier (m). Monitoring does not proceed unless this switch is turned on, i.e. set to 1. If the switch is turned off, the MONITOR package returns control to the user program without producing any output. Initially, when a program is loaded, all switches are turned off.

The switches can be of use under the following circumstances:

1 After loading a program and before entering it, the operator can set the switches according to the programmer's requirements so that only those that apply to parts of the program not already checked, and that are required for the current run, are turned on.

2 Switch settings can be changed during the running of a program, which is useful if MONITOR printing occurs during a frequently performed loop.

3 A program can be test run (for instance, during parallel running) with the MONITOR points rendered inactive by having all switches turned off.
A switch can be set in two ways:

1. By program, using the PLAN macro-instructions ON and OFF (see Chapter 4).

2. By operator action, typing the following console messages:
   
   ON #PROG n   or
   
   OFF #PROG n

   where PROG is the program name and n is the switch number.

**PLAN 1 and PLAN 2**

The #MONITOR directive cannot be used in programs written in PLAN 1 or PLAN 2. In these cases, MONITOR must be called as an ordinary subroutine. The general sequence of instructions is as follows:

```
STO 1 XXXX  [XXXX is a location in the lower storage area]

CALL 1 MONITOR
   n/m
   c/a
```

where n is the identifier and the c/a parameters are repeated n times. These parameters are dealt with fully in Chapter 6 (page 24).

**The MINIMONITOR Subroutine**

MINIMONITOR is an abbreviated version of the MONITOR subroutine that can be useful when storage is limited. MINIMONITOR is called in a way similar to that in which MONITOR is called in PLAN 1 or PLAN 2. The general sequence of instructions is:

```
STO 1 XXXX

CALL 1 MINIMONITOR
   n/m
   c/a
```

The MINIMONITOR printout commences with Mn, where n is the identifier. The contents of the eight accumulators and the contents of the specified areas are printed as follows:

1. Location address within the program as a decimal number.
2. The contents as four characters.
3. The contents as eight octal digits.
4. The contents as a signed, zero suppressed, decimal integer.

Zero words are printed. The printout of each specified area is separated from the preceding area's printout by one line space.

In the case of MINIMONITOR, the printed contents of accumulator X1 are those set up by the CALL instruction. If a printout of the contents of X1 as they were prior to the CALL instruction is required, then location XXXX must (unless it is another accumulator) be among those specified by the parameters c/a.

When the subroutine is entered it examines the switch bit of word 30 which corresponds to the first digit of the identifier n. If the switch bit is set on, the subroutine requests a line printer as LPO. If a line printer is not available, the program halts with the console message:

```
HALTED MP
```

If this occurs, the operator should make a line printer available, and type:

```
GO
```

to resume the program. If no line printer can be made available, the switches 0 to 9 should all be turned off by an appropriate console message, and the program resumed by:

```
GO
```

Chapter 9
When the program resumes, all the monitor switch bits are re-examined; so that if the switches are not all turned off and a line printer is still not available, the program will halt again with the same output message.

The core store utilization of the MINIMONITOR subroutine is 144 words. It comprises:

- 8 words of lower variable store.
- 35 words of lower preset store.
- 6 words of literals store.
- 95 words of program operation store.

**The MONITORTP Subroutine**

The MONITORTP subroutine produces MONITOR-type information by means of the paper tape punch. MONITORTP is called similarly to MONITOR (see above):

```
STO 1 XXXX
CALL 1 MONITORTP
n/m
```

The details punched for each specified word of core store are the same as those given in the MONITOR printout. There are no tabulation characters in the output, i.e. it is in a fixed format, and each line ends with a newline character. Zero words are ignored, except at the beginning of each area.

**THE TRACING FACILITIES**

The tracing facilities that can be used during program testing provide continuous information about the state of a program, unlike those for MONITOR, which provide information only at certain fixed points. Another difference is that the tracing facilities are used mainly to provide information on the actual instructions being obeyed and the path taken through the program, whereas MONITOR is normally used to give information on the state of data areas. The tracing facilities that are available for use with PLAN programs are fully described in the manual ‘PLAN Program Development Aids’.

**SYMBOL ANALYSIS FACILITIES**

In addition to the symbol analyses produced by the compiler listing and store mapping facilities, routines are available to assist the ‘debugging’ of PLAN source programs by producing a listing in alphabetical order of all symbolic names used in a segment, together with the line numbers of all statements which refer to them. This helps the programmer to see how changes in one part of a segment may affect other parts. The programs providing these facilities are fully described in the manual ‘PLAN Program Development Aids’.

**AMENDMENT OF PLAN SOURCE PROGRAMS**

The amendment of PLAN source programs on magnetic tape and cassette tape is dealt with in Chapter 10, and the amendment of PLAN source programs on disc is covered in Chapter 11. PLAN source programs on paper tape may be amended by the use of the program #XRTB, a description of which is given in the manual *PLAN Program Development Aids*. 
SEQUENCING OF PLAN COMPILER AND CONSOLIDATOR OUTPUT CARDS

The following information is given to facilitate the recognition of cards punched out by the PLAN 1 and 2 compilers, the PLAN 3 compiler #XPLE, and the consolidator #XPCC. The output cards from these programs are punched with identifying information in the last eight columns as follows:

1 Semi-compiled object program cards
   Columns 73 to 76: First four characters of the segment name.
   Columns 77 to 80: Sequence number in character form, commencing with 0000 for the name block, incremented by one per card. Each segment is sequenced separately.

   The last card of each segment has a 10 overpunch in column 80.

2 Segment leader cards
   Columns 73 to 76: First four characters of the segment name.
   Columns 77 to 80: Sequence number in character form, commencing with 0000, incremented by one per card. Each segment leader is sequenced separately.

   Column 77 has a 10 overpunch.
   The last card of each segment leader has a 10 overpunch in column 80.

3 Request slip, General Purpose Loader, and consolidated leader.
   Columns 73 to 76: Program name (in request slip and consolidated leader cards only).
   Columns 77 to 80: Sequence number in character form, commencing with 0000 for the request slip, incremented by one per card.

   Column 77 has an 11 overpunch.
   The last card of the consolidated leader has a 10 overpunch in column 80.
Chapter 10
Source Programs on Magnetic Tape and Cassette Tape

INTRODUCTION

Routines are available for creating and maintaining files of PLAN source and semi-compiled programs on magnetic tape or cassette tape, in 1900 standard subfile format. Selective recompilation of programs or segments from the files may be achieved under the control of parameters provided by the programmer. The routines have been designed to give as fast and automatic running as possible, with a minimum of operator intervention.

These routines are known collectively as the COSY compilation system.

THE COSY COMPILATION SYSTEM

The COSY compilation system comprises two alternative suites of programs, one for files on magnetic tape and the other for files on cassette tape. The system has been designed to run on the following minimum computer configuration:

- A central processor with not less than 8K words.
- One paper tape reader and/or one card reader.
- One line printer.
- One paper tape punch and/or one card punch (optional).
- Four magnetic tape decks or four cassette tape stations.

If they are available, additional magnetic tape decks or cassette tape stations may be utilized when certain of the optional facilities offered by the system are required; this would obviate the need for tape changes, and hence make the run faster and more automatic.

For files on magnetic tape a choice of three compilers is provided, one to operate in central processors with a minimum of 8K words, one to give faster running in processors with not less than 16K words, and a third to permit still faster running in processors with not less than 8K words and having exchangeable discs as backing storage.

For use with magnetic tapes, the programs that make up the COSY compilation system are:

1. XPMS, editor program to create files of PLAN source and semi-compiled programs in subfile format on magnetic tape.
2. XPMP, editor program to update files of PLAN source and semi-compiled programs in subfile format on magnetic tape.
3. XPLW, PLAN 3 batch compiler, with input and output on magnetic tape in subfile format, for use on processors with not less than 8K words.
4. XPLV, PLAN 3 batch compiler, with input and output on magnetic tape in subfile format, for use on processors with not less than 16K words.
5. XPLY, PLAN 3 batch compiler, with input and output on magnetic tape in subfile format, for use on processors with not less than 8K words and having exchangeable disc stores.

For use with cassette tapes, the programs that make up the COSY compilation system are:

1. XPMK, editor program to create files of PLAN source and semi-compiled programs in subfile format on cassette tape.
2. XPMJ, editor program to update files of PLAN source and semi-compiled programs in subfile format on cassette tape.
3  #XPLR, PLAN 3 batch compiler, with input and output on cassette tape in subfile format, for use on processors with not less than 8K words. This compiler does not consolidate, its output being in unconsolidated semi-compiled form.

4  #XPCX, batch consolidator, with input and output on cassette tape in subfile format.

The cassette tape version of COSY can be used on configurations with or without a console typewriter.

SUBFILE FORMATS

Tapes produced by the COSY editor programs are composite files, consisting of a series of subfiles. Each program presented as input to the system forms a separate subfile. These subfiles, which are at level 0, may be either simple or composite. If the program being input is entirely in PLAN source form, it is placed in a simple subfile; if it includes semi-compiled segments, it is placed in a composite subfile.

Composite subfiles comprise two simple subfiles at a lower level, level 1. The first of these contains segments of the input program in PLAN source form; the second contains segments of the input program in semi-compiled form. Any segment may be held in either or both of these forms.

Within the PLAN source subfile, whether this be at level 0 or at level 1, each segment is preceded by its individual steering information.

Subfile names

The names of the level 0 subfiles are specified by the user, by means of #IDENTITY parameters, as described on page 6. The subfile names may be up to twelve characters long, of which the first four must form a unique identification among the level 0 subfile names in the file. The subfiles are placed on the tape in alphabetical order of subfile name.

If a subfile is composite, the two level 1 subfiles are given the names NAMEySOURCEy and NAMEyS-CSEGS respectively, where NAME represents the first four characters of the level 0 subfile name.

CREATING A COSY FILE OF SOURCE AND SEMI-COMPILED PROGRAMS

Program #XPMS is used to create a file of PLAN source and semi-compiled programs on magnetic tape; program #XPMK is used to create a file of PLAN source and semi-compiled programs on cassette tape. Both programs as supplied have a priority of 80.

Hardware Requirements for #XPMS

The hardware requirements for #XPMS are:

Approximately 3K words of core store.

One paper tape reader and/or one card reader.

One line printer.

Two magnetic tape decks (one of which is for the program library tape).

Further magnetic tape decks (optional).

These are the requirements for this program only; they must be enlarged if #XPMS is regarded as part of the complete compilation system (see the section 'The COSY Compilation System' above).

The deck containing the program library tape is used only to find the program; it is closed as soon as #XPMS has been loaded.

The deck containing the output file is allocated as MT1 when the program is entered, and is closed at the end of the run.

If the new output file includes a copy of a complete pre-existing file (see under 'Initial Control Parameters' below), then a further deck containing the input file is allocated as MT2 when the program is entered, and is unloaded at the end of the run.
If individual programs are incorporated in the output file from other magnetic tape files (see under 'Programs and Control Parameters' below), then the further decks containing these input files are allocated and unloaded as required, as MT4.

**Hardware Requirements for #XPMK**

The hardware requirements for #XPMK are:

- Approximately 3K words of core store.
- One paper tape reader and/or one card reader.
- One line printer.
- Two cassette tape stations (one of which is for the program library tape).
- Further cassette tape stations (optional).

These are the requirements for this program only; they must be enlarged if #XPMK is regarded as part of the complete compilation system (see the section 'The COSY Compilation System' above).

The station containing the program library tape is used only to find the program; it is closed as soon as #XPMK has been loaded.

The station containing the output file is allocated as CT1 when the program is entered, and is closed at the end of the run.

If the new output file includes a copy of a complete pre-existing file (see under 'Initial Control Parameters' below), then a further station containing the input file is allocated as CT2 when the program is entered, and is unloaded at the end of the run.

If individual programs are incorporated in the output file from other cassette tape files, (see under 'Programs and Control Parameters' below), then the further stations containing these input files are allocated and unloaded as required, as CT3.

**Character Peripheral Usage, #XPMS and #XPMK**

An input peripheral is required to read in control parameters and new programs. The required peripheral, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions' on pages 14 and 16), and is released when either all the input data has been read or there is a change of input medium. The input medium can be changed at any time after the first two parameter lines have been read, by a #SWITCH directive. When this is encountered in the input data the present reader is released and one of the opposite type is allocated.

A line printer is allocated as LP0 when the program is entered, and is released at the end of the run.

**Input Data**

Throughout the rest of this description of COSY file creation the term 'magnetic tape' should be construed as referring to conventional magnetic tape reels or to cassette tapes, whichever is relevant to the particular program used. Details given of paper tape or card input are applicable equally to #XPMS and to #XPMK.

The input data consists of:

1. Parameters to be read from paper tape and/or cards.
2. PLAN source and/or semi-compiled programs to be inserted in the output file from paper tape, cards, and/or magnetic tape.

The input data to be read from the paper tape reader and/or card reader may be regarded as one input file comprising three parts:

1. Initial control parameters.
2. A batch of programs, each program preceded by its own set of control parameters and optionally followed by #FINISH. Control parameters for any programs to be brought in individually from magnetic tape will also appear here.

Chapter 10
#STOP as a file terminator.

Each of these three parts will now be considered in turn.

Both for the initial control parameters and for the individual program control parameters, each new parameter line starts at column one of a card or at the first character position of a paper tape line. Paper tape lines are terminated by a newline character and have a maximum permitted size of 128 internal characters: beta shift characters and delta shift characters other than $. [, ], !, -, newline and horizontal tabulation are not accepted. Paper tape lines are converted to card image format; after conversion they must not contain more than 80 characters. Any paper tape lines which contravene any of these rules are flagged with a preceding *P* in the line printer output.

**Initial Control Parameters**

The initial control parameters comprise up to four parameter lines, of which one is compulsory and the other three are optional. They define the output magnetic tape, the input magnetic tape (if a complete input magnetic tape file is being copied onto the output tape), and the compiler to be used (if the output is to be compiled immediately). These parameters may appear in any order, except for the restriction on REN described below, provided that they appear before the first #IDENTITY line (described under 'Programs and Control Parameters' below). They are based on the same generalized form as those for the PLAN compilers, as described on page 21 of Chapter 7.

The initial control parameters are as follows:

1. **OUT**

   This parameter is compulsory. It causes a magnetic tape to be opened as an output file.

   A scratch tape or a named tape may be opened for the output file.

   If a named tape is to be opened for the output file, the parameter has the following format:

   OUT(FI LE NAME(FGN/RSN), TSN).

   where FILENAME = the existing 12-character file name. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

   FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted or expressed as zero the file generation number on the tape is not checked.

   RSN = the reel sequence number, in decimal, in the range 0 to 511. The reel sequence number will always be checked, but if it is zero it may be omitted from the parameter, in which case the preceding solidus should also be omitted.

   TSN = the tape serial number, in octal, in the range octal 0 to octal 37777777. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the tape serial number will not be checked.

   If FGN is omitted, and RSN is present, the solidus must also be present; thus:

   OUT(FI LE NAME(/RSN), TSN)

   If FGN and RSN are both omitted, the related brackets are also omitted; thus:

   OUT(FI LE NAME, TSN)

   If a tape labelled SCRATCH TAPE is required to be opened as the output tape, then it should be declared in the OUT parameter with SCRATCH TAPE as its file name; if a specific tape labelled SCRATCH TAPE is required to be opened, then the tape serial number should additionally be quoted.

   If it is intended to rely on Executive's search for a tape with an expired retention period to locate an output tape, then it will be necessary to use an OUT parameter with FILENAME written as ####, thus:

   OUT(# ####)
If it is required to pick up a tape of a given serial number and an expired retention period as the output tape, and there is any uncertainty as to the existing file name, then it is permissible to use a parameter of the form:

\[ \text{OUT}(\text{##}##, \text{TSN}) \]

The sequence of events in this case is that Executive locates the lowest numbered deck (or station) containing a tape with an expired retention period, opens the tape, and relabels it SCRATCH TAPE. The editor program then proceeds to check the tape serial number. Should this not be the required one, the program unloads the tape and halts with the message "HALTED:- NEEDS TAPE NNNN" (or an equivalent message on machines without a console typewriter).

This gives the operator the opportunity to put up the correct tape, or to put lower numbered decks off-line, as may be necessary, before restarting by the console message "GO #XPM?". Any tapes opened and subsequently closed in this manner have of course had their original header labels overwritten: the facility should therefore be used with care.

2 REN

This parameter is optional; if present, it must follow immediately after the OUT parameter. It causes the magnetic tape specified in the OUT parameter to be renamed.

The parameter has the following format:

\[ \text{REN} (\text{FILENAME}(\text{FGN}/\text{RSN}, \text{RET})) \]

where FILENAME = the 12-character file name to be written to the output tape. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted the output tape is given a file generation number of zero.

RSN = the reel sequence number, in decimal, in the range 0 to 511. It may be omitted, in which case the preceding solidus should also be omitted. If it is omitted the output tape is given a reel sequence number of zero.

RET = the number of days retention period to be given to the output tape, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted, or if a number greater than 4095 is specified, then the tape is given a retention period of 4095.

If FGN is omitted, and RSN is present, the solidus must also be present; thus:

\[ \text{REN} (\text{FILENAME}(/\text{RSN}, \text{RET})) \]

If FGN and RSN are both omitted, and RET is present, the preceding comma must also be present; thus:

\[ \text{REN} (\text{FILENAME}(, \text{RET})) \]

If FGN, RSN and RET are all omitted, the related brackets are also omitted; thus:

\[ \text{REN} (\text{FILENAME}) \]

3 IN

This parameter is optional. It causes a magnetic tape to be opened as an input file. If a magnetic tape is opened by an IN parameter, then all the programs on it will be copied onto the output tape, but without steering information. The additional programs read in from the paper tape reader and/or card reader will be merged with those read from the input magnetic tape, in order to maintain the alphabetical sequence of the subfiles on the output tape.

The parameter has the following format:

\[ \text{IN} (\text{FILENAME}(\text{FGN}/\text{RSN}), \text{TSN}) \]

where the significances of FILENAME, FGN, RSN and TSN, and the rules governing their use, are as described under the OUT parameter above.
Only one IN parameter may be used in the initial control parameters. If it is desired to incorporate copies of further magnetic tape input files in the output tape, this can be achieved by use of the #FILE facility (see under 'Programs and Segments from Magnetic Tape' below).

The compiler parameter

This parameter is optional. It causes a specified program (normally a compiler) to be brought into core when the editor program deletes itself. If the compiler parameter is present, then on the completion of its run the editor program deletes itself with a DELTY 'FIND' message. The editor program provides the 'FI' portion of this message; the variable portion is obtained from the compiler parameter. The compiler parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests may be included in the parameter, in addition to the fields discussed below.

The generalized format of the compiler parameter is:

NEXT(Variable portion of DELTY 'FIND' message)

This may usually be formalized as:

NEXT(#ABCD #NAME N)

where #ABCD = the name of the program to be brought into core when the editor program deletes itself.

#NAME = the name of the appropriate search program, if applicable; for example, #TAPE or #XPK. This field is omitted where the tape from which the program is to be loaded bears the same name as the specified program; that is, if the file name on the tape is of the form PROGRAM ABCD.

N = the amount of core into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program's request block.

The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

With #XPMS, if the program specified by this parameter is a compiler whose name commences with the letters XPL, then NEXT and the brackets may be omitted, together with the first hashmark. For example, if #XPMS is to be followed by a COSY compiler in a 16K core store, the compiler parameter could appear simply as:

XPLV #TAPE

With #XPMK, if the program specified by this parameter is the COSY compiler #XPLR, then NEXT and the brackets may be omitted, together with the first hashmark. In the majority of #XPMK runs, therefore, the compiler parameter will appear simply as:

- XPLR #TAPE

Note that if #XPMK is being used in a configuration without a console typewriter, then the 'delete and find' facility will not be available, and the compiler parameter, if present, will be ignored.

Programs and Control Parameters

The initial control parameters as described above are followed by a batch of programs, each with its own preceding control parameters, and each optionally followed by #FINISH. Control parameters for programs to be brought into the output file from other magnetic tape files (other than a file declared in an IN initial control parameter) will also appear here.

The control parameters have a format similar to that of PLAN compiler directives. Where a parameter has an operand, the operand must be punched starting from character 16 of a paper tape line or column 16 of a card. In the case of paper tape, if horizontal tabulation characters are used they will be expanded to the field format of PLAN source cards.
The control parameters are as follows:

1  #IDENTITY

This parameter is compulsory. It must be the first parameter for any program. It establishes the subfile name under which a program can be identified by the COSY editor programs.

With #XPM, the operand fields of the #IDENTITY parameter are checked for invalid characters.

If any are found, the parameter is rejected - see the section 'Line Printer Error Messages'. #XPM then searches for the next #IDENTITY or the #STOP parameter.

The parameter has the following format:

#IDENTITY   IDENTIFIER, CHECKNO

where IDENTIFIER = a name which will become the name of the subfile into which the program is put. It may be up to 12 characters long, of which the first must be alphabetic, and the first four must be uniquely recognizable in the file. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters. Internal spaces will not be removed, and will be included in the count of 12 characters.

CHECKNO = up to eight alphanumeric characters; preceding and internal spaces will be removed, and will not be included in the count of eight characters.

CHECKNO may be omitted, in which case the preceding comma should also be omitted. It may be used for any purpose as required, e.g. as an account code, project number or program mark number. If present, it will be inserted in the program's request slip when the program is compiled.

Note 1: A subfile name should always be specified with a #IDENTITY parameter.

Note 2: The programs in the input file must be in alphabetical sequence of identifier.

The remaining three parameters may be input in any order.

2  #STEER

This parameter is optional, as it is not always required to compile a program when it is first put on a file. It specifies the steering information which is inserted in the source subfile in the first record of each segment of the program. When the output tape is subsequently submitted to the compiler, only those segments with steering information present will be compiled.

#STEER may be used between segments of a program, if, for example, it is desired to compile only the later segments of a program. The specified steering information will be inserted in the source subfile only for those segments read in subsequent to the reading of the #STEER parameter. Further, the parameter may be used more than once in connection with a program, with different operands, if it is desired to give different steering information to different segments.

The parameter has the following format:

#STEER   Steering information

where Steering information = LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST and/or BINARY and/or MAP(OFF) and/or FULLLIST(OFF) separated by commas. MAP implies OBJECT. FULLLIST implies MAP and OBJECT. MAP(OFF) implies OBJECT. FULLLIST(OFF) implies MAP(OFF) and OBJECT.

If it is desired to compile some of the earlier segments of a program but to omit some of the later ones, #STEER with a blank operand may be inserted between the two groups of segments, to denote that no steering information is to be inserted for the following source segments until a further #STEER is met.

Note: the MAP, FULLLIST, MAP(OFF) and FULLLIST(OFF) facilities are not available with the cassette tape version of the COSY compilation system.

With #XPM (but not with #XPMK), if it is desired to insert for a particular segment steering information which differs from that specified for a group of segments by a #STEER parameter, then a #SEGSTEER parameter may be used. The format of this parameter is as follows:

#SEGSTEER   Steering information

where Steering information is as defined for #STEER.
The effect of the #SEGSTEER parameter is that the steering information it specifies is inserted for the next following segment only. Steering information for subsequent segments reverts to that specified by the #STEER parameter previously in force. #SEGSTEER may be used similarly to insert steering information for a single segment when there is no overall steering information in force. It is permissible to use #SEGSTEER with a blank operand when it is required to inhibit the application of steering information to a particular segment.

#SEGSTEER may be used immediately before a #FILE parameter (see page 9) to insert steering information for a segment read from magnetic tape. It should be noted, however, that if #SEGSTEER is used before a #FILE parameter from which the segment name has been omitted, then the effect will be the same as that of a #STEER parameter used in the same circumstances; that is, the steering information will be applied to all the segments read from the specified subfile.

3 #SEMISTEER

This parameter is optional. With #XPMS, it causes the segments held in the semi-compiled subfile on the output tape to be consolidated when the tape is subsequently submitted to the compiler. With #XPMK it causes the segments held in the semi-compiled subfile on the output tape to be copied across to the compiler output tape when the tape is subsequently submitted to the compiler; these segments are thus made available to the consolidator. The effect of omitting this parameter, therefore, is to inhibit the inclusion of segments from the semi-compiled subfile should the program proceed to compilation and consolidation.

The parameter has the following format:

    #SEMISTEER

with the operand field blank.

4 #NEWPROG

This parameter is optional. It specifies options in the manner in which the program is to be recorded on the output tape; namely, that there are semi-compiled segments present, and/or that card sequencing of source statements is to be removed.

The parameter has the following format:

    #NEWPROG    S/C, V

where either or both of S/C and V may be present. If both S/C and V are present, they may appear in either order separated by a comma. If only one is present, the comma is not required. If neither is present, the parameter may be omitted.

S/C denotes that there are semi-compiled segments present in the input of this program. It causes the program to be put into a composite subfile instead of a simple subfile. If S/C is omitted, but semi-compiled segments are nevertheless present in the input of this program, then the semi-compiled segments will be completely ignored by the COSY editor.

V causes the two words of card sequencing to be removed from source input. The records written to the source subfile by the COSY editors are variable length records with a maximum of 21 words (the record word count in the first word, followed by up to twenty words comprising the source line), trailing spaces being removed from the source line before the record is output. With sequenced card input there are no trailing spaces, the sequence fields being in columns 73 to 80; so that the output would effectively be fixed-length 21-word records. With the V operand present, however, the card sequence fields are rendered to spaces before the trailing spaces are removed; so that shorter variable-length records can again be output. With these shorter records, faster speeds can be obtained when the output tape is subsequently read.

When #XPMK is removing card sequencing in response to a V operand on a #NEWPROG parameter, it checks first that the sequence is correct, and outputs on the line printer a list of any lines that are out of order. Any such lines printed are flagged with *P* in the left-hand margin.

#XPMS checks card sequencing and outputs on the line printer a list of any lines that are out of order, whether V was present in the operand of a #NEWPROG parameter or not. Any such lines printed are flagged with *P* in the left-hand margin.

Each set of control parameters is followed by its associated program (unless this is to be read from magnetic tape; see below), on paper tape and/or cards, in PLAN source and/or semi-compiled form. If semi-compiled segments are to be included they must be input after all the PLAN source segments. Particular segments may be included in either or both of PLAN source and semi-compiled forms. A #FINISH parameter should be input at the end of each program. It may be omitted; but it provides additional security if a file is shared between different users.
With \#XPMK the segments within any subfile containing an overlay program should normally be arranged in the following order:

1. Steering segment.
2. Overlay segments in ascending sequence of unit number within area number.
3. Permanent segments.

If this sequence is departed from, difficulties may be experienced when the program is submitted for compilation and consolidation. See the note at the end of the section 'The COSY Compilers'.

A \#SWITCH directive may be input wherever required to change the input medium from paper tape to cards, or vice versa.

**Programs and Segments from Magnetic Tape**

It should be remembered that the term 'magnetic tape' as used in this section refers to conventional magnetic tape reels or to cassette tapes, depending on which version of the COSY compilation system is being used.

All input magnetic tapes must be in subfile format. (See further at the end of this sub-section.)

If a complete magnetic tape input file is to be copied across to the output file, it may be declared in an IN parameter among the initial control parameters, as described earlier. In that case, no control parameters for individual programs from that file will appear. The following restrictions apply when a magnetic tape file is declared in an IN parameter among the initial control parameters.

1. Only one magnetic tape file can be so declared.
2. Only a complete file can be copied.
3. No steering information for the programs copied is written to the output file.
4. The file must be in the format output by the COSY editor programs.

If an individual program, or an individual segment of a program, is to be copied from an input magnetic tape; or if programs (or segments) are required to be copied from additional input magnetic tapes; or if it is desired to insert steering information, so that a program copied from an input magnetic tape can proceed immediately to compilation; then the \#FILE parameter can be used. This parameter has an associated IN parameter, similar in format to the IN initial control parameter. The pair of parameters will appear among the control parameters and/or program following a \#IDENTITY line.

With \#XPMS, the operand fields of the \#FILE parameter are checked for invalid characters. If any are found, the parameter is rejected - see the section 'Line Printer Error Messages'. \#XPMS then proceeds to the next segment or program to be written, and either rewinds the input tape if it is to be left open (that is, a \#CONTINUE or \#REWIN is present) or closes the tape.

The format of the \#FILE parameter and its associated parameters is:

```
FILE IDENTIFIER/SEGMENTNAME
IN(FILENAME(FGN/RSN), TSN)
```

optionally followed by

```
#CONTINUE
```

or

```
#REWIND
```

where IDENTIFIER = the name of the subfile to be searched for and opened on the magnetic tape specified by the following IN parameter. It may be up to 12 characters long; if it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters; internal spaces will not be removed, and will be included in the count of 12 characters. IDENTIFIER may be omitted, in which case /SEGMENTNAME must also be omitted. If it is omitted, the editor program will imply to the \#FILE line the same operand as for the associated \#IDENTITY line.
SEGMENTNAME = the name of the segment to be copied from the subfile. It may be up to 11 characters long; if it is punched with less than 11 characters the program will supply spaces at the right-hand end to complete the 11 characters. In the case of the segment name, the program will remove any internal spaces, and disregard them in the count of 11 characters (this is to conform with compiler practice). The segment name may be omitted, in which case the preceding / must also be omitted. If it is omitted, the whole of the subfile will be copied onto the output tape. The segment name should not be omitted if the input subfile contains segments in consolidated semi-compiled form.

FILENAME, FGN, RSN, TSN together specify the input magnetic tape to be opened and searched for the subfile specified by the preceding #FILE parameter. Their significance and the rules governing their use are as described under the OUT parameter on page 4.

#CONTINUE or #REWIND should be used if more than one segment or program is to be copied from the same magnetic tape. They cause the tape to be left open; in their absence, the tape would be unloaded after the program or segment specified by the #FILE parameter has been copied.

#CONTINUE should be used if the next program or segment required occurs later on the input tape. The input tape is not rewound; except that if the program reads right down the tape without finding the subfile specified by the #FILE parameter, then the tape is rewound to beginning of tape, but is left open. When #CONTINUE is used, the IN parameter following the next #FILE parameter may be omitted.

#REWIND should be used if the next program or segment required occurs earlier on the input tape. The input tape is rewound to beginning of tape, but is left open. When #REWIND is used, the IN parameter following the next #FILE parameter may be omitted. With #XPMK (but not #XPMPS) #REWIND should be used in place of #CONTINUE if the next #FILE parameter refers to the next segment in the same subfile.

Where programs or segments are being copied to the output file under the control of a #FILE parameter, the input file from which they are being copied, though it must be in subfile format, may depart from the format of the COSY editor output tapes in the following ways:

1. The subfiles need not be in alphabetical order.
2. The simple subfiles containing semi-compiled programs or segments do not have to be contained within composite subfiles.
3. The subfiles from which semi-compiled programs or segments are to be copied may contain them in consolidated semi-compiled form.

Programs or segments may therefore be copied to the output file under the control of a #FILE parameter from tapes produced by the relevant PLAN compilers and consolidators, as well as from other COSY editor output tapes.
File Terminator

The format of the file terminator for the paper tape and/or card input file is:

#STOP

Examples of an Input File

Example 1

The following paper tape and/or card file might be input to create a cassette tape file containing two programs:

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>20</th>
<th>26</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHELF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDENTITY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFFER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#PROGRAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial control parameters open a particular scratch tape and rename it.

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>20</th>
<th>26</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>SED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDENTITY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFFER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#SEQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data for first program.

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>20</th>
<th>26</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>SED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#SEQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data for second program.

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>20</th>
<th>26</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>SED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Semi-compiled segments

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>20</th>
<th>26</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINISH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

File terminator.

A #SWITCH would appear wherever necessary if the input medium changes.

Chapter 10
Example 2

The following paper tape and/or card input file might be input to create a new magnetic tape file containing all the programs from an earlier file plus an additional program, some of whose segments are obtained from a third magnetic tape file:

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT(COSY.FILE(2), 22367)</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>REN(NEW.COSY.FILE(1,1,15))</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>IN(COSY.FILE(5), 22372)</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>XPLMishments</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>SALES, SUM., S.P.</td>
<td>15.069</td>
<td>40</td>
</tr>
<tr>
<td>PSTEE</td>
<td>LIST, OBJECT, MAP</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>#PROGRAM</td>
<td>5571.0/FIRST</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>PSTEE</td>
<td>OBJECT, MAP</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td>STATISTICS/SEG2</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>INIT(infile(5))</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>#CONTINUE</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>FILE</td>
<td>STATISTICS/SEG5</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>PSTEE</td>
<td>LIST, OBJECT, MAP</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>#PROGRAM</td>
<td>5571/FOURTH</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>#FINISH</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>#STOP</td>
<td>186</td>
<td>32</td>
<td>40</td>
</tr>
</tbody>
</table>

Line Printer Output

The initial control parameters are printed on the line printer. The rest of the parameters and data from the paper tape/card input file are not printed, except that any paper tape lines that contain a format error and any cards on which a sequence error has been detected (see under #NEWPROG on page 8) are printed and flagged with *DE in the left-hand margin. While the output file is being created, a summary of the programs and segments written to it is printed on the line printer. The contents of each program's subfile are analyzed on a separate page. An example is given below. With #XPMS (but not #XPMK) each subfile's analysis may alternatively be separated from the next five line spaces, if switch 18 is set on.

After a page heading "Created by #XPMS" (or "Created by #XPMK") each segment is summarized in four columns:

1. The identifier. Note that this is the subfile name, which is not necessarily the same as the program name.

2. The segment name.

3. The number of days since last amended. For segments input from paper tape or cards this will always be 0000. For segments input from a magnetic tape file, it will be the number of days since the segment was last amended in that file.

4. The steering information present. If no steering information is present the column is left blank. For segments in semi-compiled form the letter C is printed in this column if #SEMISTEER was present; otherwise it is left blank.
If semi-compiled segments are present for a program, the start of the summary of the semi-compiled subfiles is indicated by the heading line "S/C SEGMENTS:"

At the end of each program's analysis is printed a note of the file name, generation number and tape serial number of the output tape, and the date.

The listing relative to Example 1 above would appear as follows:

```
CREATED BY #XPMK #01
ANAL. 58 ONESEG 0000 LIST

XPMKOUTPUT_0001 TAPE *22261 15/6/67

CREATED BY #XPMK #01
CASHMONEY SEG1 0000 LIST, OBJECT
CASHMONEY SEG2 0000 LIST, OBJECT

S/C SEGMENTS:-
CASHMONEY SEGX 0000 C
CASHMONEY SEGY 0000 C

XPMKOUTPUT TAPE *22261 15/6/67

CREATED BY #XPMK #01
END OF XPMKOUTPUT 0001
```

**Line Printer Error Messages**

The following error messages may be output on the line printer:

- **INVALID #IDE FOUND**
  - **SUBFILE**: subfilename
  - This message occurs in #XPMS only. An invalid character has been encountered in the operand field of the #IDENTITY parameter. The parameter is rejected and #XPMS searches for the next #IDENTITY or the #STOP parameter.

- **INVALID #FIL FOUND**
  - **SUBFILE**: subfilename
  - This message occurs in #XPMS only. An invalid character has been encountered in the operand field of a #FILE parameter. The parameter is rejected and #XPMS proceeds to the next segment or program to be written, and either rewinds the input tape if it is to be left open (that is, a #CONTINUE or a #REWIND is present) or closes the tape.

- **SUBFILE name REJECTED. OUT OF ORDER**
  - The name specified on the #IDENTITY line is out of alphabetical sequence. The program will not be written to the output file.

- **SUBFILE name NOT FOUND**
  - **SEGMENT name NOT ON INPUT FILE**
  - The name specified in a #FILE line cannot be found on an input magnetic tape. The editor program proceeds to the next segment or program to be written.

- **NO INPUT FILE OPEN FOR name**
  - A #FILE has no associated IN parameter. The editor program proceeds to the next segment or program to be written.

- **SUBFILE name NOT COMPLETE**
  - This message occurs in #XPMS only. The end-of-tape marker has been reached on the output cassette tape. The tape is closed off, but not all the source or semi-compiled records for the last subfile ('name') may have been copied across. Any remaining input will be ignored.

- **TAPE FULL. name NOT ON FILE**
  - This message occurs in #XPMS only. The end-of-tape marker has been reached on the output tape. The run finishes in the normal manner. All subfiles prior to 'name' are correctly on the output file, but 'name' and subsequent subfiles are omitted.
The #NEWPROG parameter with an S/C operand is not present for the program currently being input, but semi-compiled segments are present in the input. No further action is taken on the semi-compiled segments and the editor program proceeds to the next program to be written.

A source segment is present in the input (either in the card/paper tape input file or in the magnetic tape subfile specified by a #FILE parameter) but a semi-compiled segment has already been received for this program. No further action is taken on the source segment and the editor program proceeds to the next segment or program to be written.

A parameter has been read, but no #END for the current segment has been detected. The program infers a #END for the current segment, and proceeds.

The first of these messages may occur with #XPMS or #XPMK; the others may occur with #XPMS only. These messages accompany and amplify the console message 'HALTED:- MISREAD ON INPUT FILE'. The run is normally abandoned.

This message may occur with #XPMS only. An unrecognized sentinel, or a start-of-subfile or end-of-subfile sentinel in an incorrect position, has been read from the main input file (that is, the file specified by the initial control parameters). No more data read from this file is copied to the output file until the next correct level 0 start-of-subfile sentinel is read. The run otherwise continues normally and the output tape is created in a correct format.

Operating Instructions for #XPMS

1 Load the following magnetic tapes:
   (a) A program library tape containing #XPMS and the compiler, if any, specified in the parameters.
   (b) The output tape as specified in the parameters, with write permit ring.
   (c) Any input tapes which are required, if sufficient decks are available. If sufficient decks are not available, then input tapes may be substituted for other closed input tapes, or for the program library tape, at the appropriate times.

2 Input the message:
   FI #XPMS #NAME
   where #NAME represents the name of the appropriate search program (generally #TAPE).

3 Load the parameters and data in the paper tape reader and/or card reader.

4 If each program’s subfile’s summary is to be printed after four line spaces instead of at head of form, input the message:
   ON #XPMS 18

5 If the initial control parameters are on paper tape, activate by:
   GO #XPMS 20

   If the initial control parameters are on cards, activate by:
   GO #XPMS 21

6 The program allocates the appropriate reader and the line printer, opens and closes or unloads files as required, and creates the output file.
After completing all the amendments for each subfile, #XPMS output one of the following console messages:

(a) If all the amendments were completed successfully, and no parameter errors were detected:
   
   0#XPMS; DISPLAY:: SERV OK subfilename

(b) If any parameter errors were detected, or if any line printer messages were output for any reason:
   
   0#XPMS; DISPLAY:: SERV ER subfilename

When the run is completed the program deletes itself, with a message in the form:

(a) if no compiler parameter was present:

   0#XPMS; DELETED:: MS

(b) if a compiler parameter was present:

   0#XPMS; DELETED:: Fi #XPL? #NAME

where #XPL? represents either #XPLV or #XPLW or #XPLY, and #NAME, if present, represents the appropriate search program (generally #TAPE), as specified in the compiler parameter. The latter part of the message will appear in the same format as in the compiler parameter and may therefore include a core request.

Note: If none of the programs on the output file has steering information present, the former of these two messages will be output, whether the compiler parameter was present or not.

**EXCEPTION CONDITIONS**

The following messages may be output on the console typewriter:

1 0#XPMS; HALTED:: INCORRECT FILE PARAMETERS. FIX & RESTART

   This message is output if any of the initial control parameters is not in the correct format. The parameter should be corrected and the program restarted.

2 0#XPMS; HALTED:: NEEDS TAPE NNNNN

   This message is output if a tape serial number has been checked and found incorrect. The correct tape as specified in the appropriate parameter should be loaded, and the program resumed by:

   GO #XPMS

3 0#XPMS; HALTED:: MISREAD ON INPUT FILE

   This message is output if a misread occurs on an input file. The run is normally abandoned.

   The message may be accompanied by a message on the line printer indicating the nature of the misread (see under 'Line Printer Error Messages' above).

4 0#XPMS; DISPLAY:: TAPE FULL

   This message is output if the end-of-tape marker is detected on the output tape. (See further under 'Line Printer Error Messages' above).

5 0#XPMS; DISPLAY:: IS

   This message is output if an incorrect sentinel is read from the main input file (the file specified by the initial control parameters). It is accompanied by the line printer message 'INCORRECT SENTINEL FOUND ON INPUT'. The run continues.

6 0#XPMS; DISPLAY:: PL
   0#XPMS; UNIT n:: FIX

   This pair of messages is output if a 'paper low' condition is detected on the line printer. The operator should rectify the condition, and the run resumes when the printer is re-allocated.

Note: If the program is being run under the control of a GEORGE 1 or GEORGE 2 operating system, any HALTED message that would otherwise occur is converted into a DISPLAY.
Operating Instructions for #XPMK

1 Load the following cassette tapes:
   (a) A program library tape containing #XPMK and, if appropriate, the compiler #XPLR.
   (b) The output tape, as specified in the parameters, with write permit button.
   (c) Any input tapes which are required.
   
   Note: If the program is being run on a machine without a console typewriter, these cassettes must be loaded on stations 0, 1, and 2 (main input tape declared by IN initial control parameter) or 3 (#FILE tape) respectively.

2 Input the message:
   
   \text{FI } \#XPMK \#NAME

   where \#NAME represents the name of the appropriate search program (generally #XPKP).

3 Load the parameters and data in the paper tape reader and/or card reader.

4 If the initial control parameters are on paper tape, activate by:
   
   \text{GO 20}

   If the initial control parameters are on cards, activate by:
   
   \text{GO 21}

5 The program allocates the appropriate reader and the line printer, opens and closes or unloads files as required, and creates the output file.

6 When the run is completed the program deletes itself, with a message in the form:
   
   (a) if no compiler parameter was present, or if the program is being run on a processor without a console typewriter:
   
   \text{DELETED MK (\#2600 \#5553)}

   (b) if a compiler parameter was present, and the program is being run on a processor with console typewriter:
   
   \text{DELETED FIND \#XPLR \#NAME}

   where \#NAME, if present, represents the appropriate search program (generally #XPKP), as specified in the compiler parameter. The latter part of the message will appear in the same format as in the compiler parameter and may therefore include a core request.

   Note: If none of the programs on the output file has steering information present, the former of these two messages will be output, whether the compiler parameter was present or not.

EXCEPTION CONDITIONS

The following messages may be output on the console typewriter or, in the case of processors without a console typewriter, displayed on the lights:

1 Typewriter message: HALTED INCORRECT FILE PARAMETERS, FIX & RESTART
   Lights display: HALTED IP (\#2400 5160)
   This message is output if any of the initial control parameters is not in the correct format. The parameter should be corrected and the program restarted.

2 Typewriter message: HALTED NEEDS TAPE NNNNN
   Lights display: HALTED NT (\#2400 5664)
   This message is output if a tape serial number has been checked and found incorrect. The correct tape as specified in the appropriate parameter should be loaded, and the program resumed by:
   
   \text{GO}

3 Typewriter message: HALTED MISREAD ON INPUT FILE
   Lights display: HALTED EI (\#2400 4551)
   This message is output if a misread occurs on an input file. The run is normally abandoned.
   
   If the nature of the misread is a failure of a block count check, the message 'INCORRECT BLOCK COUNT ON UNIT n' is output on the line printer.
4 Typewriter message: DISPLAY TAPE FULL
Lights display: DISPLAY TF (#2500 6446)
This message is output if the end-of-tape marker is detected on the output tape. See further under 'Line Printer Error Messages' above.

5 Typewriter messages: DISPLAY PL
UNIT n FIX
Lights display: DISPLAY PL (#2500 6054)
These messages are output if a 'paper low' condition is detected on the line printer. The operator should rectify the condition. On machines with a console typewriter the run resumes when the printer is re-allocated. On machines without a console typewriter the run is resumed by pressing the GO button, but the message UNIT n FIX (#4000.00nn) is immediately displayed. It is then necessary to re-allocate the printer and again press the GO button.

UPDATING A COSY FILE OF SOURCE AND SEMI-COMPILED PROGRAMS

Program #XPMR is used to update a file of PLAN source and semi-compiled programs on magnetic tape; program #XPMJ is used to update a file of PLAN source and semi-compiled programs on cassette tape. Both programs as supplied have a priority of 80.

The files being updated must be in standard subfile format as described on page 2.

The updating may comprise any or all of the following:
1 The insertion of further programs in source and/or semi-compiled form.
2 The insertion of further segments in existing programs, in source and/or semi-compiled form.
3 The deletion of programs or segments.
4 The replacing of programs or segments by new versions.
5 The insertion, deletion or replacing of lines within segments.
6 The provision of new steering information for a subsequent batch compilation run.

Notes: 1 Provided that only the first of the above facilities is required, the file creation programs, #XPMS or #XPKM, may be used to update a file, in place of #XPMR or #XPMJ respectively. This would save time, as the run would not include a 'write amendment tape' phase. The file to be updated would be declared in an IN initial control parameter. Steering information may then be included on the output tape for the new programs being added only. For full details of #XPMS and #XPKM see pages 2 et seq. of this chapter.

2 The programs #XPMS or #XPKM may also be used to update a file when changes of types 2, 3, 4 or 6 above are to be made; but here the saving may be nullified by the necessity of separately declaring each of the programs or segments required to be carried forward to the updated file in #IDENTITY and #FILE parameters. The IN initial control parameter would not be used. No changes can be made within segments if #XPMS or #XPKM is used instead of the normal updating programs #XPMR or #XPMJ.

New programs or segments to be incorporated into a file may be input from paper tape or cards, or from other magnetic tape (#XPMR) or cassette tape (#XPMJ) files, provided that these also are in subfile format.

#XPMR and #XPMJ can in addition provide lists, or dumps to paper tape or cards, of source or semi-compiled programs or segments from the file, as may be required.

A second output file containing only those programs or segments to be compiled can be created if required. This file will always be named COMPILE FILE.

A maximum of 49 programs per file may be amended during any one run. There is no limit to the number of programs which may be placed on a file, other than the physical capacity of the tape.

Please note that throughout the following descriptions the term 'amendments' is to be construed as including new programs and segments to be inserted into a file.
Phases of Operation of #XPMR and #XPMJ

#XPMR and #XPMJ work in two phases.

The first phase consists of reading parameters and amendments from paper tape and/or cards, printing them, analyzing them, and writing them onto a work tape. The work tape is then rewound or realigned, and the main input and output files are opened.

The second phase merges the amendments tape and the main input file to produce the updated output file. At the same time, any programs or segments to be brought in from subsidiary input files on magnetic tape (#XPMR) or cassette tape (#XPMJ) are merged onto the output file. A list of the programs and segments on the updated output file is printed on the line printer.

It is during this phase that any source lists or source dumps requested are produced, and that COMPILER FILE, if specified, is written.
Hardware Requirements for #XPMR

The hardware requirements for #XPMR are:

- Not more than 5888 words of core store.
- One paper tape reader and/or one card reader.
- One line printer.
- One paper tape punch and/or one card punch (optional).
- Four magnetic tape decks (one of which is for the program library tape).
- Further magnetic tape decks (optional).

These are the requirements for this program only; the core store requirement must be enlarged if #XPMR is regarded as part of the complete compilation system (see the section 'The COSY Compilation System' above).

The deck containing the program library tape is used only to FIND the program; it is closed as soon as #XPMR has been loaded.

A work tape (or optionally a specified tape) is allocated as MT0 when the initial control parameters have been read. The program checks the tape serial number, and if it finds that it has been specified for one of the other output tapes, it unloads the tape and opens another. MT0 is written as an amendments tape, and is rewound when all the amendments and control parameters have been input. It is then read, and is closed at the end of the run.

The deck containing the input file is allocated as MT2 when all the parameters and amendments have been read, and is unloaded at the end of the run.

The deck containing the output file is allocated as MT1 when all the parameters and amendments have been read, and is closed or unloaded at the end of the run, depending on whether or not a compilation run is to follow.

The deck containing the tape for COMPILe FILE is allocated as MT3, if requested, when all the parameters and amendments have been read, and is closed at the end of the run.

If programs are to be incorporated into the output file from other subsidiary input files, then the further decks containing these input files are allocated and unloaded as required, as MT4.

Hardware Requirements for #XPMJ

The hardware requirements for #XPMJ are:

- Not more than 5888 words of core store.
- One paper tape reader and/or one card reader.
- One line printer.
- One paper tape punch and/or one card punch (optional).
- Four cassette tape stations (one of which is for the program library tape).
- Further cassette tape stations (optional).

These are the requirements for this program only; the core store requirement must be enlarged if #XPMJ is regarded as part of the complete compilation system (see the section 'The COSY Compilation System' above).

The station containing the program library tape is used to FIND the program, and is closed as soon as #XPMJ has been loaded. It may then be necessary to replace the library cassette by another cassette, if the COMPILe FILE or subsidiary input file facilities are used.

A work cassette (or optionally a specified cassette tape) is allocated as CT0 when the initial control parameters have been read. The program checks the tape serial number, and if it finds that it has been specified for one of the other output tapes, it unloads the tape and opens another. CT0 is written as an amendments tape, and is realigned when all the amendments and control parameters have been input. It is then read, and is closed at the end of the run.

The station containing the input file is allocated as CT2 when all the parameters and amendments have been read, and is unloaded at the end of the run.
The station containing the output file is allocated as CT1 when all the parameters and amendments have been read, and is closed or unloaded at the end of the run, depending on whether or not a compilation run is to follow.

The station containing the cassette for COMPILE FILE is allocated as CT3, if requested, when all the parameters and amendments have been read, and is closed at the end of the run.

If programs are to be incorporated into the output file from other subsidiary input files, then the further stations containing these input files are allocated and unloaded as required; as CT3 if COMPILE FILE has not been requested, or as CT4 if COMPILE FILE has been requested and sufficient stations are available.

**Character Peripheral Usage, #XPMR and #XPMJ**

An input peripheral is required to read in control parameters and amendments. The required peripheral, a paper tape reader or a card reader, is allocated initially as determined by the entry point used (see 'Operating Instructions' on pages 37 and 39), and is released when either all the input data has been read or there is a change of input medium. The input medium can be changed at any time after the first two parameter lines have been read, by a #SWITCH directive. When this is encountered in the input data the present reader is released and one of the opposite type is allocated.

A line printer is allocated as LP0 when the program is entered, and is released at the end of the run.

**Input Data**

Throughout the rest of this description of COSY file updating the term 'magnetic tape' should be construed as referring to conventional magnetic tape reels or to cassette tapes, whichever is relevant to the particular program used. Details given of paper tape or card input are applicable equally to #XPMR and to #XPMJ.

The input data consists of:

1. Parameters and amendments to be read from paper tape and/or cards.
2. A magnetic tape file of source and/or semi-compiled programs, to be updated.
3. Optionally, other magnetic tape files containing programs or segments to be inserted into the updated file.

The input data to be read from the paper tape reader and/or card reader may be regarded as one input file comprising three parts:

1. Initial control parameters.
2. A batch of amendments, the amendments for each program or segment being preceded by their own set of control parameters and optionally followed by #FINISH. Control parameters for amendments to be brought in from other magnetic tape files will also appear here.
3. A file terminator.

Each of these three parts will now be considered in turn.

Both for the initial control parameters and for the individual amendment control parameters, each new parameter line starts at column one of a card or at the first character position of a paper tape line. Paper tape lines are terminated by a newline character and have a maximum permitted size of 128 internal characters; beta shift characters and delta shift characters other than $, , , +, newlines and horizontal tabulation are not accepted. Paper tape lines are converted to card image format; after conversion they must not contain more than 80 characters. Any paper tape lines which contravene any of these rules are flagged with a preceding *P* in the line printer output.

**Initial Control Parameters**

The initial control parameters comprise two compulsory and further optional parameter lines. They define the magnetic tapes to be used for the main input and output files; optionally, the magnetic tapes to be used for COMPILE FILE and the amendments file; and the compiler to be used, if the output is to be compiled immediately. These parameters may appear in any order, except for the restriction on REN described below, provided that they appear before the first #IDENTITY line (described under 'Amendments and Control Parameters' below).
The initial control parameters are as follows:

1  IN

This parameter is compulsory. It causes a magnetic tape to be opened as the input file to be updated.

The parameter has the following format:

IN(Filename(FGN/RSN), TSN)

where Filename = the existing 12-character file name. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted or expressed as zero the file generation number on the tape is not checked.

RSN = the reel sequence number, in decimal, in the range 0 to 511. The reel sequence number will always be checked, but if it is zero it may be omitted from the parameter, in which case the preceding solidus should also be omitted.

TSN = the tape serial number, in octal, in the range octal 0 to octal 37777777. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the tape serial number will not be checked.

If FGN is omitted, and RSN is present, the solidus must also be present; thus:

IN(Filename(/RSN), TSN)

If FGN and RSN are both omitted, the related brackets are also omitted; thus:

IN(Filename, TSN)

2  OUT

This parameter is compulsory. It causes a magnetic tape to be opened for the updated output file.

A scratch tape or a named tape may be opened for the output file.

If a named tape is to be opened for the output file, the parameter has the following format:

OUT(Filename(FGN/RSN), TSN)

where the significances of Filename, FGN, RSN and TSN, and the rules governing their use, are as described under the IN parameter above.

If a tape labelled SCRATCH TAPE is required to be opened as the output tape, then it should be declared in the OUT parameter with SCRATCH TAPE as its file name; if a specific tape labelled SCRATCH TAPE is required to be opened, then the tape serial number should additionally be quoted.

If it is intended to rely on Executive's search for a tape with an expired retention period to locate an output tape, then it will be necessary to use an OUT parameter with FILENAME written as #### thus:

OUT(####)

If it is required to pick up a tape of a given serial number and an expired retention period as the output tape, and there is any uncertainty as to the existing file name, then it is permissible to use a parameter of the form:

OUT(####, TSN)

This facility should however be used with caution; see the description on page 5 of this Chapter.

3  OUTA

This parameter is optional. If entry point 22 or 23 to the editor is used, the OUTA parameter causes the specified magnetic tape to be opened as a second output file, to which will be written a copy of only those programs which are to be compiled. This tape is renamed COMPILE FILE, with a file generation number of zero, a reel sequence number of zero and a retention period of seven days.

If this parameter is absent, and entry point 22 or 23 to the editor is used, then a tape labelled SCRATCH TAPE is opened for COMPILE FILE.
If this parameter is present, but entry point 20 or 21 to the editor is used, then the parameter is accepted but ignored.

The parameter has the following format:

\[ \text{OUTA}(\text{FILENAME}(\text{FGN/RSN}), \text{TSN}) \]

where the significance of FILENAME, FGN, RSN and TSN, and the rules governing their use, are as described under the IN parameter above. If a specific tape labelled SCRATCH TAPE is required to be opened, then the OUTA parameter should be used quoting SCRATCH TAPE as the file name and specifying the tape serial number. If it is intended to rely on Executive's search for a tape with an expired retention period to locate an output tape for COMPILE FILE, then either of the forms

\[ \text{OUTA}(###) \quad \text{or} \quad \text{OUTA}(###, \text{TSN}) \]

of the parameter may be used, as for the OUT parameter above.

4 OUTB

This parameter is optional. It causes the specified magnetic tape to be opened as the amendments file. If it is omitted a tape labelled SCRATCH TAPE is opened for the amendments file.

The parameter has the following format:

\[ \text{OUTB}(\text{FILENAME}(\text{FGN/RSN}), \text{TSN}) \]

where the significances of FILENAME, FGN, RSN and TSN, and the rules governing their use, are as described under the IN parameter above. If a specific tape labelled SCRATCH TAPE is required to be opened, then the OUTB parameter should be used quoting SCRATCH TAPE as the file name and specifying the tape serial number.

If it is intended to rely on Executive's search for a tape with an expired retention period to locate a tape for the amendments file, then either of the forms

\[ \text{OUTB}(###) \quad \text{or} \quad \text{OUTB}(###, \text{TSN}) \]

of the parameter may be used, as for the OUT parameter above.

5 REN

This parameter is optional. If present, it must immediately follow an OUT or an OUTB parameter. It causes the magnetic tape opened by the preceding OUT or OUTB parameter to be renamed.

The parameter has the following format:

\[ \text{REN}(\text{FILENAME}(\text{FGN/RSN, RET})) \]

where FILENAME = the 12-character file name to be written to the tape. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted the tape is given a file generation number of zero.

RSN = the reel sequence number, in decimal, in the range 0 to 511. It may be omitted, in which case the preceding solidus should also be omitted. If it is omitted the tape is given a reel sequence number of zero.

RET = the number of days retention period to be given to the tape, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted, or if a number greater than 4095 is specified, then the tape is given a retention period of 4095.

If FGN is omitted, and RSN is present, the solidus must also be present; thus:

\[ \text{REN}(\text{FILENAME}(/\text{RSN, RET})) \]

If FGN and RSN are both omitted, and RET is present, the preceding comma must also be present; thus:

\[ \text{REN}(\text{FILENAME}(, \text{RET})) \]

If FGN, RSN and RET are all omitted, the related brackets are also omitted; thus:

\[ \text{REN}(\text{FILENAME}) \]
Two different REN parameters may be present, one following an OUT and the other following an OUTB parameter.

6 INB

This parameter is optional. It causes #XPMR or #XPMJ to be entered at the beginning of the merge phase; i.e., the input and output tapes are opened immediately, the magnetic tape specified by the INB parameter is opened as an input amendments file, and the merge is commenced without reading any amendments from paper tape or cards. This facility is useful if for some reason only the first phase of the program was completed successfully on a previous run.

The parameter has the following format:

INB(FILENAME(FGN/RSN), TSN)

where the significance of FILENAME, FGN, RSN and TSN, and the rules governing their use, are as described under the IN parameter above.

If a scratch tape was written as the amendments file, and was not renamed, it will be necessary for the INB parameter to specify SCRATCH TAPE as the header label file name, with a tape serial number for identification. It should be noted that the scratch tape indicator # # # used with parameters specifying output tapes cannot be used with the INB parameter to specify an input tape.

If INB appears among the initial control parameters there would be no amendments to follow from paper tape or cards, so the last of the initial control parameters must be followed by the file terminator #STOP.

7 The compiler parameter

This parameter is optional. It causes a specified program (normally a compiler) to be brought into core when the editor program deletes itself.

If the compiler parameter is present, then on the completion of its run the editor program deletes itself with a DELTY 'FIND' message. The editor program provides the 'FI' portion of this message; the variable portion is obtained from the compiler parameter. The compiler parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests may be included in the parameter, in addition to the fields discussed below.

The general format of the compiler parameter is:

NEXT(Variable portion of DELTY 'FIND' message)

This may usually be formalized as:

NEXT(#ABCD #NAME N)

where #ABCD = the name of the program to be brought into core when the editor program deletes itself.

#NAME = the name of the appropriate search program, if applicable; for example, #TAPE or #XKP. This field is omitted where the tape from which the program is to be loaded bears the same name as the specified program; that is, if the file name on the tape is of the form PROGRAM ABCD.

N = the amount of core into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program’s request block.

The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

With #XPMR, if the program specified by this parameter is a compiler whose name commences with the letters XPL, then NEXT and the brackets may be omitted, together with the first hashmark. For example, if #XPMR is to be followed by a COSY compiler in a 16K core store, the compiler parameter could appear simply as:

XPLV #TAPE

With #XPMJ, if the program specified by this parameter is the COSY compiler #XPLR, then NEXT and the brackets may be omitted, together with the first hashmark. In the majority of #XPMJ runs,
therefore, the compiler parameter will appear simply as:

XPLR #TAPE

Note that if #XPMJ is being used in a configuration without a console typewriter, then the 'delete and find' facility will not be available, and the compiler parameter, if present, will be ignored.

Amendments and Control Parameters

The initial control parameters as described above are followed by a batch of amendments, each with its own preceding control parameters, and each optionally followed by #FINISH. Control parameters for programs to be brought into the updated file from other magnetic tape files will also appear here.

The control parameters have a format similar to that of PLAN compiler directives. Where a parameter has an operand, the operand must be punched starting from character 16 of a paper tape line or column 16 of a card. In the case of paper tape, if horizontal tabulation characters are used they will be expanded to the field format of PLAN source cards.

With #XPMR, the operand fields of the #IDENTITY parameter are checked for invalid characters. If any are found, the parameter is flagged with an asterisk. #XPMR then searches for the next #IDENTITY or the #STOP parameter.

The control parameters are as follows:

1 #IDENTITY

This parameter is compulsory. It must be the first parameter for any amendment. It identifies the program or segment which is to be amended; or, in the case of a program new to the file, it establishes the name of the new subfile which is to be created to contain that program.

The parameter has the following format:

#IDENTITY IDENTIFIER/SEGMENTNAME, CHECKNO

where IDENTIFIER = the name of the subfile containing the program to be amended. It may be up to 12 characters long, of which the first must be alphabetic. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters. Internal spaces are not removed, and are included in the count of 12 characters. If the amendment is a program new to the file, the identifier becomes the name of the new subfile, in which case the first four characters must be uniquely recognizable in the file.

SEGMENTNAME = the name of the segment to which the amendment relates. It is necessary only if the amendment relates to a specific segment; if the amendment relates to a whole program, then SEGMENTNAME is omitted, in which case the preceding / should also be omitted. If the amendment comprises the insertion of a new segment, SEGMENTNAME should be the name of the segment after which the new one is to be inserted; but see further the discussion of #NEWSEG on page 28. (For cassette tapes, see in this connection the note at the end of the section 'The COSY Compilers'.)

If the amendment relates to a segment that has no name, then SEGMENTNAME is omitted and, except as described on pages 27 (#COPY) and 29, it is immaterial whether the solidus is present or not, SEGMENTNAME may be up to 11 characters long; if it is punched with fewer than 11 characters the program will supply spaces at the right-hand end to complete the 11 characters. In the case of the segment name, the program will remove any internal spaces, and disregard them in the count of 11 characters (this is to conform with compiler practice).

CHECKNO = up to eight alphanumeric characters, which may be used for any purpose as required, e.g. - as an account code, project number or program mark number. Preceding and internal spaces are removed, and are not included in the count of eight characters. This item does not have to be the same as the check number, if any, already existing for this subfile, because the editor program does not check the check number. If it differs, the new check number will replace the old one. It
may be omitted, in which case the preceding comma should also be omitted. If it is omitted, the check number, if any, already existing for this subfile is carried forward onto the updated file. If a check number is present when a program proceeds to compilation, it will be inserted by the compiler into the program's request slip.

It is possible to rename a subfile, provided that the first four characters of the subfile's name are not altered (the file is ordered by alphabetical sequence of the first four characters of subfile name). If it is desired to alter a subfile's name, an expanded form of the #IDENTITY parameter is used. If there is more than one amendment of a program, and it is desired to alter the subfile's name, the first #IDENTITY relating to that program must be in the expanded format; any attempt to rename a subfile by other than the first parameter relating to it will be ignored. Any subsequent amendments to the program in the same run must refer to the subfile by its old name, not by its new one.

The expanded format of the #IDENTITY parameter when renaming a subfile is as follows:

```
#IDENTITY IDENTIFIER=NEWIDENTITY/SEGMENTNAME,CHECKNO
```

where NEWIDENTITY = the new name for the subfile. It may be up to 12 characters, of which the first four must be the same as the first four characters of IDENTIFIER. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters. Internal spaces are not removed, and are included in the count of 12 characters.

The rules governing the rest of the operand are as previously stated.

Note 1: A subfile name should always be specified with a #IDENTITY parameter.

Note 2: Amendments relating to different segments of the same program must be in the sequence in which the segments appear within the program in the file. Amendments relating to different programs need not be in the sequence in which the programs appear in the file. However, considerable time will be saved if the amendments are in the same sequence as the programs appear in the file, i.e., in alphabetical sequence of subfile name (and hence of IDENTIFIER), as this will avoid the need for additional passes of the amendments tape during the merge stage.

Programs and segments on the input magnetic tape file for which no #IDENTITY parameters are present in the updating run will be copied across to the output file, but with any previously existing steering information removed.

The remaining parameters may be input in any order.

2 #STEER

This parameter is optional, as it is not always required to compile an amended program. It specifies the steering information which is inserted in the source subfile in the first record of each segment of the program, as discussed below. When the output tape is subsequently submitted to the compiler, only those segments with steering information present will be compiled.

The parameter has the following format:

```
#STEER Steering information
```

where Steering information = LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST and/or BINARY a=1/or MAP(OFF) and/or FULLLIST(OFF) separated by commas. MAP implies OBJECT.

FULLLIST implies MAP and OBJECT. MAP(OFF) implies OBJECT.

FULLLIST(OFF) implies MAP(OFF) and OBJECT.

Please refer to the note on MAP and FULLLIST on page 36 of Chapter 7. The MAP, FULLLIST, MAP(OFF) and FULLLIST(OFF) facilities are not available with the cassette tape version of the COSY compilation system.

If the #STEER parameter is used with the first amendment for a program, the steering information it specifies will be inserted for all segments of the program (including any prior to the segment, if any, specified in the associated #IDENTITY line), unless overridden for particular segments by #SEGSTEER parameters (described below), or until a further #STEER is encountered in a later amendment for the same program.

If the #STEER parameter is used with an amendment other than the first for a program, the steering information it specifies will be inserted for the segment specified in the associated
#IDENTITY line, and for all subsequent segments, unless overridden for particular segments by #SEGSTEER parameters, or until a further #STEER is encountered in a later amendment for the same program. The steering information will also be inserted for all segments between that specified on the associated #IDENTITY line and that specified on the preceding #IDENTITY line, but not for the latter segment nor for any prior segments.

#STEER may be used with more than one amendment to a program. As indicated above, the steering information specified by each #STEER parameter will be applicable to the program until the next #STEER is encountered.

If it is desired to compile some of the earlier segments of a program but to omit some of the later ones, #STEER may be used with a blank operand to denote that no steering information is to be inserted in segments determined in the above manner, until a further #STEER is met.

Whether or not #STEER is present, any previously existing steering information is removed from the file when the file is updated. If it is desired to proceed after an updating run to recompile a program which has not been amended, this can be achieved by including in the paper tape or card input file a dummy amendment consisting solely of the #IDENTITY, #STEER and #FINISH parameters.

3 #SEGSTEER

This parameter is optional. It specifies the steering information which is to be inserted in the source subfile for the particular segment being amended, where this differs from the overall steering information as determined by #SEGSTEER, or where no overall steering information is provided. This facility is useful when listing or monitoring of amended segments only is required.

If the amendment is the insertion of a new segment, then the steering information is inserted for the new segment, not for the segment referenced on the #IDENTITY line.

The parameter has the following format:

```
#SEGSTEER  Steering information
```

where Steering information = LIST or SHORTLIST and/or OBJECT and/or MONITOR and/or MAP and/or FULLLIST and/or BINARY and/or MAP(OFF) and/or FULLLIST(OFF) separated by commas. MAP implies OBJECT. FULLLIST implies MAP and OBJECT. MAP(OFF) implies OBJECT. FULLLIST(OFF) implies MAP(OFF) and OBJECT.

Please refer to the note on MAP and FULLLIST on page 36 of Chapter 7. The MAP, FULLLIST, MAP(OFF) and FULLLIST(OFF) facilities are not available with the cassette tape version of the COSY compilation system.

#SEGSTEER may be used with a blank operand, to denote that no steering information is to be inserted for the particular segment.

4 #SEMISTEER

This parameter is optional. With #XPMR, it causes the segments held in the semi-compiled subfile on the output tape to be consolidated when the tape is subsequently submitted to the compiler. With #XPMRJ it causes the segments held in the semi-compiled subfile on the output tape to be copied across to the compiler output tape when the tape is subsequently submitted to the compiler; these segments are thus made available to the consolidator. The effect of omitting this parameter, therefore, is to inhibit the inclusion of segments from the semi-compiled subfile should the program proceed to compilation and consolidation.

The parameter has the following format:

```
#SEMISTEER
```

with the operand field blank.

If it is required, it may appear in any amendment to a program.

Note: For non-overlay programs, where a segment appears in both a program's source subfile and semi-compiled subfile, if during an updating run an OBJECT steering line is applied to the source segment and #SEMISTEER is also applied to the program, it will be the recompiled segment from the source subfile which is subsequently accepted for consolidation, and the segment of the same name from the semi-compiled subfile will be rejected. For overlay programs, no segment should be presented to the compiler in both its source and semi-compiled form.
This parameter is optional. It causes a dump of the program or segment to be given on the line printer and/or the paper tape punch or card punch. If a dump of a whole program is called for, both the source and the semi-compiled subfiles are dumped. Individual segment dumps can be obtained from either subfile.

The parameter has the following format:

```
#COPY     p1, p2
```

where p1, p2 = the symbolic names of the required peripherals (LP, TP or CP). TP and CP cannot be specified together, but either can be specified in conjunction with LP; or any of the three devices can be specified alone. If only one device is specified, p2 and the comma are omitted.

If a source dump on the card punch is called for, the editor program generates fresh card sequencing, in hundreds (i.e. starting with a one in column 78).

If the #COPY parameter is used in conjunction with amendment types #REPROG or #REPSEG, then a dump both of the program or segment being replaced and of the new one replacing it will be given. If only one of these dumps is required, it may be achieved by programming two separate amendments instead, a #DELPLOY or #DELSG followed by a #NEWPROG or #NEWSEG (as these are separate amendments each must have its own #IDENTITY); #COPY being associated with the former or the latter only, depending on whether a separate record of the old or the new program or segment is required.

If the #COPY parameter is used in conjunction with amendment types #LINEDIT or #COMMENT a dump of the segment in its amended form is given.

If the #COPY parameter is used in a dummy amendment, that is, one from which the amendment type parameter is omitted, then #XPMR gives a dump of the whole program or of one segment according to whether the solidus is present in or absent from the operand field of the #IDENTITY parameter; a dump of an unnamed segment may thus be obtained by including the solidus in the #IDENTITY parameter but omitting any segment name. If the #COPY parameter is used in a dummy amendment with #XPMJ, then a dump of the whole program or of one segment is given according to whether the segment name is present in or is absent from the #IDENTITY parameter.

Amendment type

This parameter is optional. It is normally present, except where a dummy amendment is being used in order to insert steering information for an unchanged program or segment, or to obtain a source dump of an unaltered program or segment. If it is absent the program or segment specified by the associated #IDENTITY line (if present on the main input file) is simply copied into the output file.

The amendment type will be one of the following:

```
#NEWPROG,    #DELPLOY,    #REPPROG,
#NEWSEG,      #DELSG,    #REPSEG,
#LINEDIT,     #COMMENT.
```

Only one of these parameters may be associated with any particular #IDENTITY line. (There is one exception to this rule; see page 32.)

#NEWPROG, #DELPLOY and #REPPROG respectively adds a new program, deletes an existing program or replaces an existing program on the file.

#NEWSEG, #DELSG and #REPSEG respectively adds, deletes or replaces a segment of an existing program.

#LINEDIT introduces detailed amendments to a segment. Lines of the segment are referenced by line number, as shown on compilation lists. Lines may be inserted, deleted or replaced.

#COMMENT is used when only the comment in a segment needs amending.

Details of each of these 'amendment type' parameters follow:
# NEWPROG
This causes a new program to be inserted in the file. The subfile containing it will be placed on the output file in alphabetical sequence of subfile name.

The parameter has the following format:

```
#NEWPROG   S/C, V
```

where both S/C and V are optional. If both are present they may appear in either order, separated by a comma. If only one (or neither) is present, the comma is not required.

S/C denotes that there are semi-compiled segments present in the input of this program. It causes the program to be put into a composite subfile instead of a simple subfile. If S/C is omitted, but semi-compiled segments are nevertheless present in the input of this program, then the semi-compiled segments will be completely ignored by the COSY editor.

V causes the two words of card sequencing to be removed from source input. This permits the source program to be entered on the file in shorter, variable-length records (see page 8 of this chapter). When #XPMJ is removing card sequencing in response to a V operand, it checks first that the sequence is correct, and outputs on the line printer a list of any lines that are out of order.

#XPMR checks card sequencing and outputs on the line printer a list of any lines that are out of order, whether V was present in the operand of the #NEWPROG parameter or not. Both with #XPMR and with #XPMJ, any such lines printed are flagged with "P" in the left-hand margin.

If #NEWPROG is present, the last of the control parameters for this amendment is followed by the new program (unless the new program is being brought in from another magnetic tape file; see 'Programs and Segments from other Magnetic Tape Files' below), on paper tape and/or cards, in PLAN source and/or semi-compiled form. If semi-compiled segments are to be included they must be input after all the PLAN source segments of the program. Particular segments may be included in either or both of PLAN source and semi-compiled forms.

#DELPROG
This causes the deletion of the complete subfile specified in the preceding #IDENTITY parameter.

The parameter has the following format:

```
#DELPROG
```

with the operand field blank.

#REPPROG
This causes the whole of a program's subfile to be replaced by a new one.

The parameter has the following format:

```
#REPPROG   S/C, V
```

where both S/C and V are optional. Their effects and the rules governing their use are as described under #NEWPROG above.

As the complete subfile is replaced when this parameter is used, it follows that it is possible for a composite subfile to be replaced by a simple subfile, and vice versa, as well as for a subfile to be replaced by one of the same type.

The COSY file updating programs will accept this parameter in either of the two forms #REPPROG or #REPROG.

If this parameter is present, the last of the control parameters for this amendment is followed by the replacement program (unless the replacement program is being brought in from another magnetic tape file; see 'Programs and Segments from other Magnetic Tape Files' below), the same conditions applying as stated above for new programs under #NEWPROG.

#NEWSEG
This causes a new segment or new segments to be inserted in an existing program's subfile.
The parameter has the following format:

```
#NEWSEG    S/C, V
```

where both S/C and V are optional. If both are present they may appear in either order, separated by a comma. If only one (or neither) is present the comma is not required.

S/C denotes that there is semi-compiled input present in this amendment. If the existing subfile is a simple subfile, S/C causes it to be converted into a composite subfile. If the existing subfile is already a composite one, S/C may be omitted. If the existing subfile is a simple one, and S/C is omitted, but semi-compiled input is nevertheless submitted, then the semi-compiled segments will be completely ignored by the COSY editor.

V causes the two words of card sequencing to be removed from source input, as discussed under "#NEWPROG" above. It should be noted here, however, that the card sequencing is only removed from the new source segments; segments already on the file are unaffected. Out of order lines are printed on the line printer in the same way as with "#NEWPROG".

If "#NEWSEG" is present, the last of the control parameters for this amendment is followed by one or more new segments (unless a new segment is being brought in from another magnetic tape file; see 'Programs and Segments from other Magnetic Tape Files' below), the same conditions applying as stated above for new programs under "#NEWPROG".

The new segment(s) will be inserted in the file immediately following the segment specified in the "#IDENTITY" parameter. With "#XPMJ", if no segment name is specified in the "#IDENTITY" parameter, and this is the first amendment of the program to occur this run, then the new segment(s) will be inserted as the first segment(s) in the appropriate subfile. If no segment name is specified in the "#IDENTITY" parameter, but this is not the first amendment of the program to occur this run, then the new segment(s) will be inserted following the segment last amended (or, if the last amendment was to the source subfile and the new segment is semi-compiled, as the first segment in the semi-compiled subfile). With "#XPMR", if no segment name is specified in the "#IDENTITY" parameter and no solidus follows the subfile name, then the segment(s) will be inserted in the same manner as with "#XPMJ"; but if a solidus follows the subfile name in the "#IDENTITY" parameter and no segment name is specified, then the new segment(s) will be inserted following a segment with no name. If in these latter circumstances no unnamed segment is encountered before the end of the subfile is reached, the error message '

`<1> SEGMENT spaces NOT FOUND ON INPUT FILE`' is output on the line printer. With "#XPMR", to insert a new segment immediately following an unamended segment that has no name it is necessary to include a dummy amendment consisting solely of a "#IDENTITY" parameter (with the segment name omitted) and a "#FINISH" parameter, immediately preceding the "#NEWSEG" amendment (whose "#IDENTITY" parameter also has no segment name present).

It should be noted that if a new segment is to be inserted immediately following a segment which has been amended then the segment name together with the preceding solidus must be omitted from the "#IDENTITY" parameter for the new segment; otherwise the editor program, having dealt with the previous amendment, will proceed down the input tape to the end of the subfile, looking again for the segment name, and will eventually bring up the line printer error message '

`<1> SEGMENT name NOT FOUND ON INPUT FILE`'.

The new segments may include both source and semi-compiled segments, provided that they can be recorded consecutively in the composite subfile; i.e., provided that they comprise the last segments of the source subfile and the first segments of the semi-compiled subfile.

For cassette tape files, see further the note at the end of the section 'The COSY Compilers'.

"#DELSSEG"

This causes a segment or segments to be deleted from a program's subfile.

The parameter has the following format:

```
#DELSSEG    N
```

where N = the number of segments to be deleted, in decimal, in the range 1 to 511. It may be omitted. If it is omitted, one segment will be deleted.

The segments deleted will commence with the segment specified in the "#IDENTITY" line.

If a segment appears in both the source and semi-compiled subfiles, from which of these it will be deleted depends on what previous amendments have occurred to the same program during the run. Amendments to different segments of a program must be in the same sequence as the segments themselves appear in the subfile; so that in the above circumstances, if a previous amendment was to a
later segment in the source subfile than the one whose name appears on the #IDENTITY line, then it will be the segment in the semi-compiled subfile which is deleted; otherwise, it will be the segment from the source subfile which gets deleted. It follows that if, in these circumstances, there are no previous amendments to the program during the run, and it is desired to delete the segment from the semi-compiled subfile, it is necessary to include a prior dummy amendment for the same (or a later) segment in the source subfile. This dummy amendment could consist solely of the #IDENTITY and #FINISH parameters.

The count of segments to be deleted, specified by N, may, if required, extend from segments in the source subfile, over consecutive segments in the semi-compiled subfile.

#REPSEG
This causes the whole of a segment or segments to be replaced by a new segment or segments.

The parameter has the following format:

#REPSEG N,S/C,V

where each of N,S/C and V is optional. N is described under #DELSERG, S/C and V are as described under #NEWSEG. If more than one of these fields are present, they may be in any order, separated by a comma. Commas are only required to separate those fields which are present.

The effect of #REPSEG is that of a #DELSERG followed by a #NEWSEG.

If #REPSEG is present, the last of the control parameters for this amendment is followed by one or more replacement segments (unless a replacement segment is being brought in from another magnetic tape file; see 'Programs and Segments from other Magnetic Tape Files' below), the same conditions applying as stated above for new programs under #NEWPROG.

The number of replacement segments need not be the same as the number of segments being replaced.

#LINEDIT
This introduces a set of detailed amendments to a source program segment.

The parameter has the following format:

#LINEDIT V

where V is optional. The effect of V is described below.

If #LINEDIT is present, the last of the control parameters for this amendment is followed by a set of detailed amendments, on paper tape and/or cards. Each detailed amendment is specified by a #ALTER line. Source lines to be deleted, or to be preceded by insertions, are referenced by the line number, starting with 1 for the first #PROGRAM line, as in the compilation lists.

Where lines are to be deleted, the #ALTER line can have either of the following formats:

#ALTER A,N

#ALTER A-B

where A = the line number of the first line to be deleted.

N = the number of lines to be deleted.

B = the line number of the last line to be deleted.

The #ALTER line will be followed by any lines which are to replace the deleted lines, in normal PLAN source form. All lines following the #ALTER line will be inserted, until the next #ALTER, #FINISH or #IDENTITY, or the file terminator, is encountered.

Where lines are to be inserted, without any deletions being made, the #ALTER line has the following format:

#ALTER A

where A = the line number of the line in front of which the new lines are to be inserted.

The #ALTER line will, as before, be followed by the new lines in normal PLAN source form.
Examples:
1  #ALTER  50
   This causes the lines following #ALTER to be inserted, the first new line becoming line 50.
2  #ALTER  50, 11
   This causes the eleven lines starting with line 50 to be deleted, and any new lines following #ALTER to be inserted in their place.
3  #ALTER  50-60
   This has exactly the same effect as Example 2.

The line numbers specified in the operand fields of successive #ALTER lines must be in rising sequence within the segment.

V in the operand field of #LINEDIT causes the two words of card sequencing to be removed from all source lines for this segment. No check is performed on the correctness of card sequencing.

It is not possible to insert or delete #END line by a #LINEDIT amendment. If an attempt is made to insert one, it will be printed on the line printer and flagged with an asterisk, but will be otherwise ignored. If the lines specified for deletion by a #ALTER line include a #END line, the program will delete all the specified lines up to but not including the #END, and will ignore any insertion lines that may follow the #ALTER.

If the #FILE facility is being used to insert a new segment (see under 'Programs and Segments from other Magnetic Tape Files', below), the segment may be amended as it is added to the file. To do this, a #LINEDIT (or #COMMENT) parameter is used in conjunction with a #NEWSEG or #REPSEG parameter, as applicable. The detailed amendments must follow immediately after the #FILE group of parameters.

#COMMENT

This introduces a set of amendments affecting only the comments on PLAN source lines.

The parameter has the following format:

#COMMENT   V

where V is optional. The effect of V is described below.

If #COMMENT is present, the last of the control parameters for this amendment is followed by a set of detailed amendments, on paper tape and/or cards. Each detailed amendment is specified by a #ALTER line. Source lines affected are referenced by line number, starting with 1 for the first #PROGRAM line, as in the compilation lists.

Where comment is to be deleted from source lines, the #ALTER line can have either of the following formats:

#ALTER   A, N
#ALTER   A-B

where A = the line number of the first line from which comment is to be deleted.
N = the number of lines from which comment is to be deleted.
B = the line number of the last line from which comment is to be deleted.

If the comment removed from these lines is to be replaced by new comment, then the #ALTER line may be followed by lines containing the new comment, in the format described below. The comment from these lines will be incorporated into the successive source lines, starting with line A. If there are more new comment lines than lines from which comment has been deleted, the excess comment lines will overwrite the comment portion of lines B + 1 et seq on the source subfile. However, it is not likely that the above formats will be used in these latter circumstances, for if it is desired to overwrite comment rather than deleting it, the following format may be used:

#ALTER   A

where A = the line number of the first of the lines whose comment portions are to be overwritten by the comment portions of the lines following the #ALTER line.
A series of new comment lines should follow, in the format described below.

The lines containing the new comment must start with a left-hand square bracket in character position 36 of a paper tape line or column 36 of a card. The comment to be deleted or overwritten must start with a left-hand square bracket in position 36 or after; if it starts after position 36, all character positions between position 35 and the bracket must be spaces. If a line to be amended contains comment starting before position 36, or an operand extending beyond position 35, the comment will not be deleted or overwritten. Any new comment line intended for such a source subfile line will be ignored.

The line numbers specified in the operand fields of successive #ALTER lines must be in rising sequence within the segment.

V in the operand field of #COMMENT causes the two words of card sequencing to be removed from all source lines for this segment. No check is performed on the correctness of card sequencing.

#COMMENT may, like #LINEDIT, be used in conjunction with the #FILE facility.

A #FINISH parameter should be input at the end of each complete amendment. This parameter makes a pair with #IDENTITY, defining a complete amendment to one segment or program. It may be omitted; but it provides additional security if a file is shared between different users.

A #SWITCH directive may be input wherever required to change the input medium from paper tape to cards, or vice versa.

Programs and Segments from other Magnetic Tape Files

It should be remembered that the term 'magnetic tape' as used in this section refers to conventional magnetic tape reels or to cassette tapes, depending on which version of the COSY compilation system is being used.

New or replacement programs, or new or replacement segments for existing programs, may be brought into the updated file from other magnetic tape files, provided that these also are in subfile format. This is done by specifying the subfile or the segment which is to be incorporated in a #FILE parameter, and specifying the magnetic tape file from which it is to be copied in an associated INA parameter. These parameters appear in conjunction with a #NEWPROG, a #REPROG, a #NEWSEG or a #REFSEG, as appropriate.

With #XPMR, the operand fields of the #FILE parameter are checked for invalid characters. If any are found, the parameter is flagged with an asterisk. #XPMR continues to read the parameters but ignores the amendment associated with the invalid #FILE.

The format of the #FILE parameter and its associated parameters is:

#FILE
IDENTIFIER/SEGMENTNAME
INA(FILENAME(FGN/RSN), TSN)

optionally followed by

#CONTINUE

or

#REWIND

where

IDENTIFIER = the name of the subfile to be searched for and opened on the magnetic tape specified by the following INA parameter. It may be omitted, in which case /SEGMENTNAME must also be omitted. If it is omitted, the editor program will imply to the #FILE line the same operand as for the associated #IDENTITY line.

SEGMENTNAME = the name of the segment to be copied from the subfile. It may be omitted, in which case the preceding / must also be omitted. If it is omitted, the whole of the subfile will be copied onto the output tape. The segment name should not be omitted if the input subfile contains segments in consolidated semi-compiled form.

FILENAME
FGN
RSN
TSN
together specify the input magnetic tape to be opened and searched for the subfile specified by the preceding #FILE parameter. Their significance and the rules governing their use are as described under the IN parameter on page 21.

Punching rules for IDENTIFIER and SEGMENTNAME are the same as for when they are used in a #IDENTITY parameter line.

Chapter 10
##CONTINUE or #REWIND should be used if more than one segment or program is to be copied from the same magnetic tape. They cause the tape to be left open; in their absence, the tape would be unloaded after the program or segment specified by the #FILE parameter has been copied.

##CONTINUE should be used if the next program or segment required occurs later on the input tape. The input tape is not rewound; except that if the program reads right down the tape without finding the subfile specified by the #FILE parameter, then the tape is rewound to beginning of tape, but is left open. When #CONTINUE is used, the INA parameter following the next #FILE parameter may be omitted.

#REWIND should be used if the next program or segment required occurs earlier on the input tape. The input tape is rewound to beginning of tape, but is left open. When #REWIND is used, the INA parameter following the next #FILE parameter may be omitted.

With #XPMJ (but not #XPMR) #REWIND should be used in place of #CONTINUE if the next #FILE parameter refers to the next segment in the same subfile.

#FILE and its associated parameters must, whenever used, come after any other of the control parameters specified above; that is, they must be the last parameters for their amendment.

Each use of #FILE constitutes a separate amendment, and so requires its own #IDENTITY.

If the #FILE facility is being used to insert a new segment, that is, in conjunction with #NEWSEG or #REPSEG (but not in conjunction with #NEWPROG or #REPPROG), then the new segment may be amended as it is added to the file. To do this a #LINEDIT or a #COMMENT is used in addition to #NEWSEG or #REPSEG. The #FILE group of parameters would then be followed by detailed amendment lines, as discussed earlier under #LINEDIT and #COMMENT.

**Example:**

Suppose a segment is to be brought in from another magnetic tape file, and incorporated as the second segment of a program on the updated file. Inappropriate comment is to be removed from certain lines of the added segment. The whole program will subsequently be compiled, and a list is required of the new segment only. Further segments will be required from the subsidiary input tape for later amendments this run. The amendment might be programmed thus:

```
<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Acc</th>
<th>1235</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>40</th>
<th>44</th>
<th>48</th>
<th>52</th>
<th>56</th>
<th>60</th>
<th>64</th>
<th>68</th>
<th>Proc. Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>#IDENTITY</td>
<td>QUAEREND/SEGONE, MARKA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#STEER</td>
<td>OBJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#SEGSTEER</td>
<td>LIST, OBJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#NEWSEG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#COMMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FILE</td>
<td>STANDARDSEG/VALIDATE3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INA(SEGFILEONE)</td>
<td>(5), 750.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#REWIND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#ALTER</td>
<td>75-77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#ALTER</td>
<td>126-132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FINISH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Where programs or segments are being copied to the output file under the control of a #FILE parameter, the input file from which they are being copied, though it must be in subfile format, may depart from the format of the COSY editor output tapes in the following ways:

1. The subfiles need not be in alphabetical order.
2. The simple subfiles containing semi-compiled programs or segments do not have to be contained within composite subfiles.
3. The subfiles from which semi-compiled programs or segments are to be copied may contain them in consolidated semi-compiled form. In this case, each segment to be copied must be specified as a separate amendment; a complete program held in semi-compiled form cannot be copied as a single amendment.

Programs or segments may therefore be copied to the output file under the control of #FILE parameters from tapes produced by the relevant PLAN compilers and consolidators, as well as from other COSY editor output tapes.
File Terminator

The format of the file terminator for the paper tape and/or card input file is normally:

#STOP

When the COMPILE FILE facility is being used, however, a parameter of the form

#TERMINATE

is required to bring the run to an end. This will replace #STOP if only one file is being updated. If a batch of files, each with its own initial control parameters and file terminator, is being updated, #TERMINATE will replace #STOP for the last file of the batch only. See further below under 'The COMPILE FILE Facility'.

The COMPILE FILE Facility

If entry point 22 (first parameters on paper tape) or 23 (first parameters on cards) to the editor program is used, instead of entry point 20 or 21 respectively, then in addition to opening an output tape for the updated file the editor program will open a second output tape, to which will be written a copy of those programs and segments which are to be compiled. A scratch tape or a named tape may be opened for this purpose depending on whether or not an OUTA parameter is present (see page 21); but the tape is always renamed COMPILE FILE, with file generation number and reel sequence number zero, and a seven day retention period. On proceeding to compilation, entry point 22 to the compiler will enable COMPILE FILE to be opened as the input file without having to read a parameter line from the paper tape reader or card reader.

The output to be written to the COMPILE FILE tape is determined as follows:

1. If a subfile is referenced in a #IDENTITY parameter and a #STEER or a #SEGSTEER parameter is associated with the #IDENTITY parameter, then all the source segments of the program contained in that subfile are copied to the COMPILE FILE tape.

2. If a subfile is referenced in a #IDENTITY parameter and a #SEMISTEER parameter is associated with the #IDENTITY parameter, then all the source segments and all the semi-compiled segments of the program held in that subfile are copied to the COMPILE FILE tape.

As the subfiles written to the COMPILE FILE tape are identical copies of the equivalent subfiles in the updated output file, the steering information for the individual segments will be the same as that in the main output file.

There are two major circumstances in which this COMPILE FILE facility should be used: when only a small proportion of a file is to be compiled, and for multi-file updating.

If a file contains a large number of programs of which only a small proportion (say 50% of the file or less) is to be compiled, considerable time is wasted by the compiler in searching through those programs for which no compilation is required. If the environment permits, therefore, the COMPILE FILE facility should be used in these circumstances.

MULTI-FILE UPDATING

The COSY system will not cater automatically for multi-reel files; multi-reel files may be dealt with as a series of single-reel files having the same file name but different reel sequence numbers, declared and updated by separate amendment files each with its own initial control parameters and file terminator. However, multi-reel files regarded in this way, and indeed a series of totally unrelated COSY files, may be processed in a single batch-updating run, using the COMPILE FILE facility.

If the series of paper tape and/or card input files is batched on the reader(s) and entry point 22 or 23 is used, then as soon as the first magnetic tape file has been updated the program will close off the various tapes (except the COMPILE FILE tape) and look for the magnetic tapes specified for the second file; and so on. If sufficient decks or stations are available for the subsequent tapes to be already set up, this processing is continuous, without operator intervention; but pauses for tape changes cannot be avoided with cassette tapes on a machine without a console typewriter, as in these circumstances the input and output tapes have to be on specified stations.
All programs or segments with steering information present, from all the updated files of the batch, will additionally be copied onto the same COMPILE FILE, so that the ensuing batch compilation run will have all the programs for recompilation, from whichever source file they originated, on one input tape.

When COMPILE FILE is used, whether for batch or single file updating, the last paper tape/card input file read must have #TERMINATE in place of #STOP as a file terminator.

**Line Printer Output**

During the 'write amendments tape' phase, the parameters and amendments read in from the paper tape reader and/or card reader are printed on the line printer. Parameters and detailed amendments are printed in full, but for new programs or new segments only the first line of the new material is printed, as the source list may optionally be obtained by use of the #COPY parameter. Any incorrect parameter lines, or any insertion lines for which the appropriate preceding parameter was omitted, will be flagged by an asterisk in the left-hand margin. Any paper tape input lines which contain a format error are printed and flagged with *P* in the left-hand margin. Card input lines in which sequence errors have been detected (see the discussions of #NEWPROG and #NEWSEG on pages 28 and 29) are also printed and flagged with *P* in the left-hand margin. Narrative may be included on any input line, in the same manner as on PLAN source statements, and if included will also be printed: it will not be written to the amendments tape unless it forms part of a PLAN source line.

As the output tape is created, a summary listing of the programs and segments in the updated file is produced on the line printer, in a similar format to that produced by the COSY file creation programs. (See the illustration on page 13.) With #XPMM (but not #XPMP) if switch 18 is set on, each subfile's analysis will be separated from the previous one by five line spaces, instead of by a throw to a new page. The third column of the segment summary shows the number of days since the subfile containing the program was last amended; it will therefore show the same figure for all segments within the subfile.

Also at this stage any source lists requested by #COPY parameters will be printed.

Certain error messages may be output on the line printer during this phase.

**Line Printer Error Messages**

The following messages may be output:

<:> SUBFILE name NOT FOUND

The name specified in a #IDENTITY line cannot be found on the input tape. The editor program proceeds to the next amendment.

<:> SEGMENT name NOT FOUND ON INPUT FILE

The user is trying to add a new subfile having a name of which the first four characters are the same as those of an existing subfile. The editor program rejects the insertion and proceeds to the next amendment.

<:> DUPLICATE name REJECTED

<:> SEMI-COMPILED CANNOT BE EDITED

The user is trying to amend a semi-compiled segment using #LINEDIT or #COMMENT. The editor program rejects this amendment and proceeds to the next one.

<:> AMENDMENT ATTEMPTED AFTER #END

A #ALTER refers to a line number beyond the end of the segment, or attempts to delete a #END line. The editor program rejects this and all subsequent amendments to that segment.

<:> NO INPUT FILE OPEN FOR name

The user is attempting to insert a new subfile or segment using a #FILE parameter, and either the IN parameter is missing or the operator has typed a GO #XPM? 29 message. The editor program proceeds to the next amendment.

<:> TAPE FULL, FOLLOWING SUBFILES NOT ON OUTPUT FILE

The end of tape marker has been detected on the output tape. The message is followed by a list of all the subfiles on the input file which have not
been copied across to the output file; the subfile which was partially written when the end of tape marker was detected has been erased from the output tape and is the one whose name appears first in the list; all previous subfiles have been correctly copied. The run terminates in the normal manner. If the output file is #XPMR or #XPMJ run, the following message will be output to the line printer at the end of that run.

The end-of-tape marker was detected when this file was created; some of the subfiles present on the previous generation may therefore be absent from this file.

Either

(a) the amendment type parameter is #NEWPROG or #REPPROG, without an S/C operand, or

(b) the amendment type parameter is #NEWSEG or #REPSEG, without an S/C operand, and the subfile specified on the #IDENTITY line is a simple subfile,

and semi-compiled segments are present in the input for this amendment. No further action is taken on the semi-compiled segments, and the editor program proceeds to the next amendment.

A source segment is present in the input (either in the card/paper tape input file or in the magnetic tape subfile specified by a #FILE parameter) but a semi-compiled segment has already been received for this program. No further action is taken on the source segment and the editor program proceeds to the next segment or program to be written.

The end-of-tape marker has been detected on the COMPILE FILE tape. The subfile which was partially written when the end-of-tape marker was detected has been erased, the trailer label has been written and the tape has been closed. The run continues until the end of the current main input and output files, at which stage the editor program deletes itself.

The first of these messages may occur with #XPMR or #XPMJ; the others may occur with #XPMR only. These messages accompany and amplify the console message 'HALTED:- MISREAD ON INPUT FILE'. The run is normally abandoned.

This message may occur with #XPMR only. An unrecognized sentinel, or a start-of-subfile or end-of-subfile sentinel in an incorrect position, has been read from the file being updated. No more data read from this file is copied to the output file.
TOO MANY SEGMENTS BEING DELETED.

This message may occur with #XPMR only. The number in the operand of a #DELSEG or a #REPSEG parameter is greater than the number of segments from the one specified in the associated #IDENTITY parameter to the end of the subfile. All the segments from the one specified to the end of the subfile are deleted.

The amendment type is #NEWPROG or #REPPROG or #NEWSEG or #REPSEG, and a #FINISH, a #STOP, or a #IDENTITY has been read, but no #END for the current segment has been detected. The program infers a #END for the current segment, and proceeds.

Operating Instructions for #XPMR

1 Load the following magnetic tapes:
   (a) A program library tape containing #XPMR and the compiler, if any, specified in the parameters.
   (b) The tape, with write permit ring, required for the amendments tape.
   (c) The input file to be updated.
   (d) The tape, with write permit ring, required for the output file, and if required:
   (e) A tape, with write permit ring, for COMPILE FILE,
   (f) Any subsidiary input files,
if sufficient decks are available. If sufficient decks are not available, then these tapes may be substituted for the program library tape, or for other closed input tapes, at the appropriate times. The tapes will be opened in the above sequence and for maximum efficiency should be loaded in that order.

2 Input the message:

   \[FI \#XPMR \#NAME\]

where \#NAME represents the name of the appropriate search program (generally #TAPE).

3 Load the parameters and data in the paper tape reader and/or card reader.

4 If each program's subfile's summary is to be printed after five line spaces instead of at head of form, input the message:

   \[ON \#XPMR 18\]

5 Activate by one of the following messages:

   If initial control parameters are on paper tape, and COMPILE FILE is not required:

   \[GO \#XPMR 20\]

   If initial control parameters are on cards, and COMPILE FILE is not required:

   \[GO \#XPMR 21\]

   If initial control parameters are on paper tape, and COMPILE FILE is required:

   \[GO \#XPMR 22\]
If initial control parameters are on cards, and COMPILE FILE is required:

GO #XPMR 23

6 The program allocates the appropriate reader and the line printer, opens and closes or unloads files as required, and creates the output file(s).

7 After completing all the amendments for each subfile, #XPMR outputs one of the following console messages:
   (a) If all the amendments were completed successfully and no parameter errors were detected:
      0#XPMR; DISPLAY:- SERV OK subfilename
   (b) If any parameter errors were detected, or if any line printer messages were output for any reason:
      # XPMR; DISPLAY:- SERV ER subfilename

8 When the run is completed the program deletes itself, with a message in the form:
   (a) if no compiler parameter was present:
      0#XPMR; DELETED:- MR
   (b) if a compiler parameter was present:
      0#XPMR; DELETED:- FI #XPL? #NAME

   where #XPL? represents either #XPLV or #XPLW or #XPLY, and #NAME, if present, represents the appropriate search program (generally #TAPE), as specified in the compiler parameter. The latter part of the message will appear in the same format as in the compiler parameter and may therefore include a core request.

Note: If none of the programs on the output file has steering information present, the former of these two messages will be output, whether the compiler parameter was present or not.

EXCEPTION CONDITIONS

The following messages may be output on the console typewriter:

1 0#XPMR; HALTED:- INCORRECT FILE PARAMETERS. FIX & RESTART.

   This message is output if any of the initial control parameters is not in the correct format, or if a REN appears other than immediately after an OUT or OUTB parameter. The parameter should be corrected and the program restarted.

   If correct parameters cannot be made available, either abandon the job or, if the message occurs for the second or a subsequent updating file in a batch updating run, type:

   GO #XPMR 27

   which will close COMPIL FILE, terminate the run and find the compiler.

2 0#XPMR; HALTED:- NEEDS TAPE NNNNN

   This message, where NNNNN represents the tape serial number in octal, is output if a tape serial number has been checked and found incorrect. The correct tape as specified in the appropriate parameter should be loaded, and the program resumed by:

   GO #XPMR

3 0#XPMR; DISPLAY:- NO MORE AMENDMENTS CAN BE ACCEPTED

   This message is output if a user seeks to amend a 50th program. There is a limit of 49 to the number of programs that can be amended in one run. The run will continue, but only the first 49 of the programs purporting to be amended will be amended.

4 0#XPMR; DISPLAY:- TAPE FULL

   This message is output if the end-of-tape marker is detected on the output tape. The message '<!> TAPE FULL. FOLLOWING SUBFILES NOT ON OUTPUT FILE' will be output on the line printer, followed by a list of subfile names. The run will terminate in the normal manner, but the subfiles whose names are listed on the line printer will not be on the output tape.
5 0#XPMR; HALTED:- MISREAD ON INPUT FILE

This message is output if a misread occurs on an input file. The run is normally abandoned. The message may be accompanied by a message on the line printer indicating the nature of the misread (see under 'Line Printer Error Messages' above).

6 0#XPMR; UNIT n :- FAIL

This message, where n is the absolute number of a paper tape reader or card reader, is the Executive message indicating either an unerasable fault on the reader concerned or a fault in the input medium which is preventing its being correctly read. If the former is the case, the run is normally abandoned. If the latter is the case, typing:

GO #XPMR 28

will permit the reader to continue with the next amendment. All amendments to the program on which the failure occurred will be ignored, but other amendments will be performed normally.

7 0#XPMR; LOAD filename

This message indicates:

(a) that a tape specified in connection with a #FILE parameter cannot be found, or
(b) that a tape specified in one of the initial control parameters for the second or a subsequent file in a batch updating run cannot be found.

If the tape can be made available, make it available and resume by:

GO #XPMR

If in case (a) above the tape cannot be made available, type:

GO #XPMR 29

The message '>: NO INPUT FILE OPEN FOR name' will be printed on the line printer, and #XPMR will proceed to the next amendment.

If in case (b) above the tape cannot be made available, type:

GO #XPMR 27

which will close COMPILE FILE, terminate the run and find the compiler.

8 0#XPMR; DISPLAY:- COMPILE FILE FULL

This message is output if the end-of-tape marker is detected on the COMPILE FILE tape. A similar message is output on the line printer. The COMPILE FILE tape is closed. The run will continue until the end of the current main input and output files, at which stage #XPMR will delete itself.

9 #XPMR; DISPLAY:- IS

This message is output if an incorrect sentinel is read from the main input file. It is accompanied by the line printer message '<!> INCORRECT SENTINEL FOUND ON INPUT'. The run continues.

10 0#XPMR; DISPLAY:- PL

0#XPMR; UNIT n :- FIX

This pair of messages is output if a 'paper low' condition is detected on the line printer. The operator should rectify the condition, and the run resumes when the printer is re-allocated.

Note: If the program is being run under the control of a GEORGE 1 or GEORGE 2 operating system, any HALTED message that would otherwise occur is converted into a DISPLAY.

Operating Instructions for #XPMJ

1 Load the following cassette tapes:

(a) A program library tape containing #XPMJ and, if appropriate, the compiler #XPLR.
(b) The tape, with write permit button, required for the amendments tape.
(c) The input file to be updated.
(d) The tape, with write permit button, required for the output file.

and if required:

Chapter 10
(e) A tape, with write permit button, for COMPILER FILE,

(f) Any subsidiary input files,

if sufficient stations are available. If sufficient stations are not available, then these tapes may be substituted for the program library tape, or for other closed input tapes, at the appropriate times.

The tapes will be opened in the above sequence, and for maximum efficiency should be loaded in that order.

Note: If the program is being run on a machine without a console typewriter, the cassettes must be loaded on the stations whose station numbers correspond to the program’s relative number for the cassettes (see under ‘Hardware Requirements for #XPMJ’, on page 19); e.g., the tape required for the amendments tape must go on station 0.

2 Input the message:
   
   `FI #XPMJ #NAME`
   
   where #NAME represents the name of the appropriate search program (generally #XPKP).

3 Load the parameters and data in the paper tape reader and/or card reader.

4 Activate by one of the following messages:

   If initial control parameters are on paper tape, and COMPILER FILE is not required:
   
   `GO 20`

   If initial control parameters are on cards, and COMPILER FILE is not required:
   
   `GO 21`

   If initial control parameters are on paper tape, and COMPILER FILE is required:
   
   `GO 22`

   If initial control parameters are on cards, and COMPILER FILE is required:
   
   `GO 23`

5 The program allocates the appropriate reader and the line printer, opens and closes or unloads files as required, and creates the output file(s).

6 When the run is completed the program deletes itself, with a message in the form:

   (a) if no compiler parameter was present, or if the program is being run on a processor without a console typewriter:

   `DELETED MJ (#2800 5552)`

   (b) if a compiler parameter was present, and the program is being run on a processor with console typewriter:

   `DELETED FIND #XPLR #NAME`

   where #NAME, if present, represents the appropriate search program (generally #XPKP), as specified in the compiler parameter. The latter part of the message will appear in the same format as in the compiler parameter and may therefore include a core request.

   Note: if none of the programs on the output file has steering information present, the former of these two messages will be output, whether the compiler parameter was present or not.

EXCEPTION CONDITIONS

The following messages may be output on the console typewriter or, in the case of processors without a console typewriter, displayed on the lights:

1 Typewriter message: HALTED INCORRECT FILE PARAMETERS, FIX & RESTART

   Lights display: HALTED IP (#2400 5160)

   This message is output if any of the initial control parameters is not in correct format, or if a REN appears other than immediately after an OUT or an OUTB parameter. The parameter should be corrected and the program restarted.
If correct parameters cannot be made available, either abandon the job or, if the message occurs
for the second or a subsequent updating file in a batch updating run, input:

GO 27

which will close COMPILE FILE, terminate the run and (on machines with a console typewriter)
find the compiler.

2 Typewriter message: HALTED NEEDS TAPE NNNNN

Lights display: HALTED NT (#2400 5664)

This message, where NNNNN represents the tape serial number in octal, is output if a tape serial
number has been checked and found incorrect. The correct tape as specified in the appropriate
parameter should be loaded, and the program resumed by:

GO

3 Typewriter message: DISPLAY NO MORE AMENDMENTS CAN BE ACCEPTED

Lights display: DISPLAY NM (#2500 5655)

This message is output if a user seeks to amend a 50th program. There is a limit of 49 to the
number of programs that can be amended in one run. The run will continue, but only the first 49
of the programs purporting to be amended will be amended.

4 Typewriter message: DISPLAY TAPE FULL

Lights display: DISPLAY TF (#2500 6446)

This message is output if the end-of-tape marker is detected on the output tape. The message '<>:TAPE FULL. FOLLOWING SUBFILES NOT ON OUTPUT FILE' will be output on the line printer,
followed by a list of subfile names. The run will terminate in the normal manner, but the subfiles
whose names are listed on the line printer will not be on the output tape.

5 Typewriter message: HALTED MISREAD ON INPUT FILE

Lights display: HALTED EI (#2400 4551)

This message is output if a misread occurs on an input file. The run is normally abandoned. If
the nature of the misread is a failure of a block count check, the message '<>: INCORRECT BLOCK
COUNT ON UNIT n' is output on the line printer.

This message is output if a misread occurs on an input file. The run is normally abandoned.

6 Typewriter message: UNIT n ERR

Lights display: #4100 00nn

This message, where n is the absolute number of a paper tape reader or card reader, is the
Executive message indicating either a fault on the reader concerned or a fault in the input medium
which is preventing its being correctly read. If the former is the case, after two or three attempts
to clear the fault, the run is normally abandoned. If the latter is the case, input:

GO 28

to permit the reader to continue with the next amendment. All amendments to the program on which
the failure occurred will be ignored, but other amendments will be performed normally.

7 Typewriter message: LDC filename

Lights display: #4200 00nn

This message indicates

(a) that a tape specified in connection with a #FILE parameter cannot be found, or

(b) that a tape specified in one of the initial control parameters for the second or a subsequent
file in a batch updating run cannot be found.

If the cassette can be made available, make it available and resume by:

GO

If in case (a) above the tape cannot be made available, input:

GO 29
The message '<colon>: NO INPUT FILE OPEN for name' will be printed on the line printer, and #XPMJ will proceed to the next amendment.

If in case (b) above the tape cannot be made available, input:

```
GO 27
```

which will close COMPILE FILE, terminate the run and (on machines with a console typewriter) find the compiler.

8 Typewriter message: DISPLAY COMPILE FILE FULL
Lights display: DISPLAY CF (#2500 4346)

This message is output if the end-of-tape marker is detected on the COMPILE FILE tape. A similar message is output on the line printer. The COMPILE FILE tape is closed. The run will continue until the end of the current main input and output files, at which stage #XPMJ will delete itself.

9 Typewriter messages: DISPLAY PL
Lights display: DISPLAY PL (#2500 6054)

These messages are output if a 'paper low' condition is detected on the line printer. The operator should rectify the condition. On machines with a console typewriter the run resumes when the printer is re-allocated. On machines without a console typewriter the run is resumed by pressing the GO button, but the message 'UNIT n FIX' (#4000 000m) is immediately displayed. It is then necessary to re-allocate the printer and again press the GO button.

THE COSY COMPILERS

It has already been stated that the compilers which form part of the COSY compilation system are:

- #XPLW, for use on configurations with magnetic tape decks and not less than 8K words of core store.
- #XPLV, for use on configurations with magnetic tape decks and not less than 16K words of core store.
- #XPLY, for use on configurations with magnetic tape decks and exchangeable disc stores and not less than 8K words of core store.
- #XPLR, for use on configurations with cassette tape units and not less than 8K words of core store.

The specifications and operating instructions for these compilers are to be found in Chapter 7, together with those for the other PLAN compilers.

The COSY compilers receive input from magnetic tape (#XPLR, cassette tape) and output onto magnetic tape (#XPLR, cassette tape). It may be necessary to provide parameters on paper tape or cards to specify the magnetic tapes or cassette tapes to be used. The format and use of these parameters are described in Chapter 7, pages 21 et seq.

**Note:**

#XPLR only: this compiler does not consolidate, and so a cassette tape COSY compilation must be followed by a #XPCX consolidation run.

If the program being consolidated is an overlay program, the segments must be presented to #XPCX in the following order:

- **(a)** Steering segment
- **(b)** Overlay segments in ascending sequence of unit number within area number.
- **(c)** Permanent segments.

It follows that the COSY editor program must present the segments to the compiler, #XPLR, in a similar order.

No library subroutine may be specified to be an overlay segment, though it may be called from an overlay segment.
Chapter 11  Source Programs on Disc

INTRODUCTION

Routines are available for creating and maintaining files of PLAN source and semi-compiled programs on exchangeable disc or fixed disc, in 1900 standard subfile format. Throughout this chapter the term "exchangeable disc" is to be construed as including twin exchangeable discs. The recompilation of programs or segments from a file may be achieved under the control of parameters provided by the programmer. The routines have been designed to give as fast and economic running as possible, with a minimum operator intervention.

These routines are known collectively as the DISC COSY compilation system.

THE DISC COSY COMPILATION SYSTEM

The DISC COSY compilation system has been designed to run on the following minimum 1900 series computer configuration:

A central processor with not less than 8K words.
One paper tape reader and/or one card reader.
One line printer.
One paper tape punch and/or one card punch (optional).
One or more magnetic tape decks (optional).
Two exchangeable disc stores or one fixed disc store.
Additional exchangeable disc stores (optional).

The programs that comprise the DISC COSY compilation system are:

1  #XPMY, editor program to create a file of PLAN source and semi-compiled programs on exchangeable disc or on fixed disc.
2  #XPMZ, editor program to update a file of PLAN source and semi-compiled programs on exchangeable disc or on fixed disc, for use in processors with not less than 16K words of core store.
3  #XPMX, editor program to update a file of PLAN source programs on exchangeable disc only, for use on processors with not less than 8K words of core store.
4  #XPMQ, utility program for obtaining security dumps of segments or programs on a DISC COSY file, for use on processors with not less than 8K words of core store. This program would not normally be used in processors with more than 16K words of core store, as the facilities it provides are included in #XPMZ.
5  #XPLX, PLAN 2 compiler with input and output on exchangeable disc or on fixed disc, for use in processors with not less than 8K words of core store. This compiler does not consolidate, its output being in unconsolidated semi-compiled form.
6  #XPLZ, PLAN 3 compiler with input and output on exchangeable disc or on fixed disc, for use in processors with not less than 16K words of core store. This compiler does not consolidate, its output being in unconsolidated semi-compiled form.
7  #XPLT, PLAN 4 compiler with input and output on exchangeable disc or on fixed disc, for use in processors with not less than 32K words of core store. This compiler does not consolidate, its output being in unconsolidated semi-compiled form.
8  #XPCL, consolidator with input and output on exchangeable disc or on fixed disc, for use in processors with not less than 8K words of core store.
9  #XPCX, consolidator with input and output on exchangeable disc or on fixed disc, for use in processors with not less than 16K words of core store.
FILE FORMATS

The files produced by the DISC COSY editor programs are composite files each containing up to 33 programs. Each program is held in a composite subfile, comprising a number of simple subfiles each containing one segment in either PLAN source or semi-compiled form. Any segment may be held in either or both of PLAN source and semi-compiled forms. All DISC COSY files use 1-block buckets.

It is not necessary for the general user to know the structure of the DISC COSY files in any greater detail. For those who require it, some additional information is given in a separate section at the end of this chapter.

When files are being initialized by the appropriate file allocator program, it is not necessary to provide the housekeeping system parameters.

CREATING A DISC COSY FILE

Program #XPMY is used to create a file of PLAN source and semi-compiled programs on disc. The program as supplied has a priority of 80.

Hardware Requirements for #XPMY

The hardware requirements for #XPMY are:

- 5K words of core store.
- One paper tape reader and/or one card reader.
- One line printer.
- One or more magnetic tape decks (optional).

and either:

- One exchangeable disc store.
- One or two further exchangeable disc stores (optional).

or

- One or more fixed disc stores.

If exchangeable disc stores are used, the file from which #XPMY was loaded, the output file and, where appropriate (see below), a work file may be on the same cartridge or on different cartridges. It is only in the latter circumstances that the optional additional exchangeable disc stores are required. Similarly, if fixed disc stores are used, these files may be on the same unit or on different units. The file containing #XPMY is closed as soon as the loading process has been completed. The other two files are specified by parameters (see under 'Input Data' below), which also determine the type of device on which the output file and the work file are to be opened. These two files must be opened on the same type of device, but #XPMY may be loaded from a device of the other type.

PLAN source and semi-compiled segments can be input from paper tape, cards and/or magnetic tape. It is also possible to input from paper tape or cards a dump of amendments subfiles produced by #XPMZ or #XPMQ. This is a file security measure, to facilitate the re-creation of a DISC COSY file in its updated form should this become necessary; it is discussed separately in the section 'Re-creating a DISC COSY File', commencing on page 41. The work file is required only if a file is being re-created.

If the file created by #XPMY is to be subsequently updated using the 8K editor program, #XPMX, it must be created on exchangeable disc, and must not include any semi-compiled segments.

If during a #XPMY create run, a subfile is set up containing no PLAN source or semi-compiled segments, the resultant output file will not be in subfile format. The effect of using such a file as input to #XPMZ is indeterminate.

Basic Peripheral Usage, #XPMY

An input peripheral is required to read in control parameters and program segments. The required peripheral, a paper tape reader or a card reader, is allocated initially as determined by the choice of entry point (see 'Operating Instructions' on page 12), and is released when either all the input data has been read or there is a change of input medium. The input medium can be changed at any time by a #SWITCH directive. When this is encountered in the input data the present reader is released and one of the opposite type is allocated.

A line printer is allocated as LPO when the program is entered, and is released at the end of the run.
Magnetic Device Usage, #XPMY

The medium used, exchangeable disc or fixed disc, is specified by the parameters, as described under ‘Input Data’ below.

OBLIGATORY DISC FILE

The output file is opened as ED0 or FD0.

OPTIONAL DISC FILE

If it is necessary to open a file as a work file, the file is opened as ED1 or FD1.

OPTIONAL MAGNETIC TAPE

If PLAN source and/or semi-compiled segments are to be copied from magnetic tape, the tape is opened as MT0.

Input Data

The input data consists of:

1. Parameters to be read from paper tape and/or cards.

2. PLAN source and/or semi-compiled segments to be inserted in the output file from paper tape, cards and/or magnetic tape.

The input data to be read from the paper tape reader and/or card reader may be regarded as one input file comprising three parts:

1. Initial control parameters.

2. A batch of programs, each program preceded by its own set of control parameters and optionally followed by #FINISH. Control parameters for any programs or segments to be brought in from magnetic tape will also appear here.

3. A batch terminator, and optionally, parameters for automatic compilation.

Each of these three parts will now be considered in turn.

Both for the initial control parameters and for the individual program control parameters, each parameter line starts at the first character position of a paper tape line or at column 1 of a card. Paper tape lines are terminated by a newline character and have a maximum permitted size of 128 internal characters: beta shift characters or delta shift characters other than $, ] , +, -, newline and horizontal tabulation are not accepted. Paper tape lines are converted to card image format; after conversion they must not contain more than 80 characters. Any paper tape lines which contravene any of these rules are flagged with a preceding "P" in the line printer output.

Initial Control Parameters

The initial control parameters comprise up to four parameter lines, of which one is compulsory and the other three are optional. They may appear in any order, provided that they appear before the first #IDENTITY line (described under ‘Programs and Control Parameters’ below). They are based on the same generalized form as those for the PLAN compilers, as described on page 21 of Chapter 7. The first two parameters described have alternative forms, depending on whether the output file is to be on exchangeable disc or on fixed disc.

The initial control parameters are as follows:
OUTE or OUTF or APPE or APPF

This parameter is compulsory. OUTE or APPE defines the exchangeable disc file to be opened for output; OUTF or APPF defines the fixed disc file to be opened for output. If OUTE or OUTF is used, the contents of the file will be overwritten by the editor program's output. If APPE or APPF is used, the editor program's output will be appended to the existing contents of the file, which will be preserved; the existing contents must be in the format produced by the DISC COSY editor programs.

The parameter has the following format:

OUTE(FILENAME(FGN1=FGN2),CSN) or OUTF(FILENAME(FGN1=FGN2))

or APPE(FILENAME(FGN1=FGN2),CSN) or APPF(FILENAME(FGN1=FGN2))

where FILENAME = the existing 12-character file name of a permanent file which is to be opened to receive the editor program's output. If the file name is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN1 = the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on line will be opened.

FGN2 = the new file generation number to be written to the output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding = sign should also be omitted. If it is omitted the file generation number will be left unchanged.

CSN = the serial number, in octal, in the range octal 0 to octal 777777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.

If FGN1 and FGN2 are both omitted the associated brackets are also omitted; thus:

OUTE(FILENAME,CSN) or OUTF(FILENAME)

or APPE(FILENAME,CSN) or APPF(FILENAME)

The version number of the file will be set to zero if FGN2 is present, or will be increased by one if FGN2 is omitted.

RENE or RENF

This parameter is optional. RENE causes the file specified in the OUTE or APPE parameter to be renamed; RENF causes the file specified in the OUTF or APPF parameter to be renamed. A RENE parameter must not be used in conjunction with an OUTF or APPF parameter, nor must a RENF parameter and an OUTE or APPE parameter be used in combination.

The parameter has the following format:

RENE(FILENAME(FGN/VN)) or RENF(FILENAME(FGN/VN))

where FILENAME = the new file name which is to overwrite the existing file name on the output file. It may be up to 12 characters long. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, the output file will be given a generation number of zero. FGN must not be expressed as -1.

VN = the version number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the solidus should also be omitted. If it is omitted, the output file will be given a version number of zero.
If FGN is omitted, and VN is present, the solidus must also be present; thus:

RENE(FILENAME(/VN)) or RENF(FILENAME(/VN))

If FGN and VN are both omitted the associated brackets are also omitted; thus:

RENE(FILENAME) or RENF(FILENAME)

If a RENE or RENF parameter is present, the file generation number and version number to be written to the output file is determined solely by that parameter; if, when RENE or RENF is present, the OUTE, OUTF, APPE or APPF parameter specifies FGN2, that field of the OUTE, OUTF, APPE or APPF parameter will be overridden.

3. OUTW

This parameter is applicable only to a file re-creation run (see page 41), and is optional in those circumstances. It defines the disc file to be opened as the work file; this must be on the same type of disc (exchangeable or fixed) as the output file. If it is omitted a scratch file will be opened on the same cartridge or unit as the output file.

If it desired to open a permanent file as the work file, the parameter has the following format:

OUTW(FILENAME(FGN1=FGN2),CSN)

where

FILENAME = the existing 12-character file name of a permanent file which is to be opened and overwritten by the editor program’s work file. If the file name is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

The significances of FGN1, FGN2 and CSN, and the rules governing their use, are as described under OUTE or OUTF above. CSN is not applicable to fixed discs, and should always be omitted, together with its preceding comma, when a file on that medium is being specified.

If it is desired to open a scratch file as the work file, but on a different cartridge or unit from the output file, then the file name is expressed as #####, and the cartridge serial number or unit serial number is quoted; thus:

OUTW(#####,CSN) or OUTW(#####,USN)

where

USN = the unit serial number, in octal, in the range octal 1 to octal 777777, of the fixed disc unit containing the file to be opened. It may be punched with or without a preceding # or * sign.

4. The compiler parameter

This parameter is optional. It causes a specified program (normally a compiler) to be brought into core when the editor program deletes itself. If the compiler parameter is present, then on the completion of its run the editor program deletes itself with a DELTY 'FIND' message. The editor program provides the 'FI' portion of this message; the variable portion is obtained from the compiler parameter. The compiler parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests may be included in the parameter, in addition to the fields discussed below.

The generalized format of the compiler parameter is:

NEXT(Variable portion of DELTY 'FIND' message)

This may usually be formalized as:

NEXT(#)ABCN #NAME N)

where

#ABC = the name of the program to be brought into core when the editor program deletes itself.

NAME = the last four characters of the name of the file containing #ABC. The name of this file must be of the form PROGRAM#NAME. #NAME may be omitted where NAME is the same as ABCD.

N = the amount of core into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program's request block.
The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

If the program specified by this parameter is a PLAN compiler with a name of the form #XPL-, then NEXT and the brackets may be omitted, together with the first hashmark, so that the parameter becomes:

\[ \text{XPL- } \text{#NAME N} \]

where \text{#NAME} and \text{N} are as defined above.

Note: The medium on which the output file and the work file are opened is determined by whichever is first read of an OUTE, an OUTF, an APPE, an APPF, a RENE or a RENF parameter.

Programs and Control Parameters

The initial control parameters as described above are followed by a batch of programs, each with its own preceding control parameters, and each optionally followed by \text{#FINISH}. Control parameters for programs or segments to be brought into the output file from magnetic tape files will also appear here.

The control parameters have a format similar to that of PLAN compiler directives. Where a parameter has an operand, the operand must be punched starting from character 16 of a paper tape line or column 16 of a card. In the case of paper tape, if horizontal tabulation characters are used they will be expanded to the field format of PLAN source cards.

The control parameters are as follows:

1 \text{#IDENTITY}

This parameter is compulsory. It must precede each program input, and must be the first parameter for each program. It defines the name to be given to the subfile containing the program in the editor program's output file.

The parameter has the following format:

\[
\text{#IDENTITY SUBFILENAME}
\]

where \text{SUBFILENAME} = a name which will become the name of the subfile into which the program is put. It may be up to 12 characters long, of which the first must be alphabetic. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

As the number of programs that can be accommodated in the output file is restricted to 33, the number of \text{#IDENTITY} parameters that are present in the basic peripheral input should not be greater than 33; and if the output file is opened by an APPE or APPF parameter, then this maximum should be reduced by the number of programs already on the file.

The four following parameters are all optional and may be input in any order before the first segment or between segments.

2 \text{#VARIABLE}

This parameter is optional. It specifies that the card sequence field is to be removed from each source record for all following source segments of this program, before the records are written to the file.

The source records written by the DISC COSY editors are variable length records, trailing spaces being removed from the source lines before the records are output. With sequenced card input there are no trailing spaces, the sequence fields being in columns 73 to 80; so that the output would effectively be fixed length 21-word records (the record word count in the first word, followed by twenty words comprising the source line). The removal of card sequencing can therefore have the effect of making each record very much shorter, so that more records can be put into each buffer. This means that reading and writing the file will be speeded up.

The parameter has the following format:

\[
\text{#VARIABLE}
\]

with the operand field blank.
When #XPMY is removing card sequencing in response to a #VARIABLE parameter, it checks first that the sequence is correct, and outputs on the line printer a list of any lines that are out of order, flagging the lines with a preceding "P". The records are written to the file, however, in the sequence in which they were input.

3 #STEER

This parameter is optional, as it is not always required to compile a program when it is first put on a file. It specifies the steering information which is to be inserted for each following source segment of this program. When the program is subsequently submitted to the compiler only those segments with steering information present will be compiled.

The parameter has the following format:

```
#STEER Steering information
```

where Steering information = any valid steering line for the compiler which is to be used.

#STEER may be used between segments where, for example, it is desired to compile only the later segments of a program. Further, the parameter may be used more than once, with different operands, if it is desired to give different steering information to different segments; each #STEER is operative only until the next #STEER is met.

If it is desired to compile some of the earlier segments of a program but to omit some of the later ones, #STEER may be used with a blank operand, to denote that no steering information is to be inserted for the following source segments until a further #STEER is met.

4 #SEGSTEER

This parameter is optional. It specifies steering information to be inserted in the next following segment only, so that steering information for subsequent segments will revert to that specified by the #STEER parameter previously in force.

The parameter has the following format:

```
#SEGSTEER Steering information
```

where Steering information is as defined for #STEER.

#SEGSTEER may be used to insert steering information for a single segment even if there is no overall steering in force. It is permissible to use #SEGSTEER with a blank operand when it is required to inhibit the application of steering information to a particular segment.

#SEGSTEER may be used immediately before a #MFILE parameter (see page 8) to insert steering information for the PLAN source segment or segments read from magnetic tape.

5 #SEMISTEER

This parameter is optional. It specifies whether or not the following semi-compiled segments of this program are to be copied across to the compiler output file when the editor output file is subsequently submitted to the compiler; and thus whether or not these segments are to be made available to the consolidator.

The parameter has the following formats:

```
#SEMISTEER CONSOLIDATE
```

or

```
#SEMISTEER
```

with the operand field blank.

If none of the semi-compiled segments in the subfile is to be consolidated, the parameter should be omitted. If all are to be consolidated, the CONSOLIDATE form should be input anywhere before the first semi-compiled segment. If only some of the semi-compiled segments are to be consolidated the parameter may be used between segments as many times as may be necessary, the CONSOLIDATE form indicating that the following semi-compiled segments are to be consolidated, and the blank operand form indicating that the following semi-compiled segments are not to be consolidated. Each #SEMISTEER is operative until the next one is met or until the end of the program's subfile.

Each set of control parameters is followed by its associated program (unless this is to be read from magnetic tape; see below), on paper tape and/or cards, in PLAN source and/or semi-compiled form. Particular segments may be
included in either or both of PLAN source and semi-compiled forms, and different versions of a segment having the same name may be input if desired. A #FINISH parameter should be input at the end of each program. It may be omitted; but it provides additional security if a file is shared between different users.

A #SWITCH directive may be input wherever required to change the input medium from paper tape to cards, or vice versa.

**Batch terminator**

The batch of programs and control parameters must be terminated by one of the following parameters:

- #STOP
- #CHAIN

with the operand blank.

#CHAIN introduces a set of parameters for automatic compilation (see below).

#STOP acts as an input file terminator.

The batch terminator causes the editor program to release the paper tape reader or card reader and to enter its end routines, in the course of which the line printer is released, all files are closed, and the program deletes itself, finding the successor program if one was specified by a compiler parameter.

**AUTOMATIC COMPIlATION**

If a #XPMY run is to be followed immediately by a compilation run using #XPLX or #XPLZ, it is possible to give #XPMY extra parameters which will cause the compiler to be entered automatically with no operator intervention.

To use this facility the normal batch terminator, #STOP should be replaced by a #CHAIN parameter, and this should be followed by the parameters for #XPLX or #XPLZ, in their normal format, and terminated by a PEND parameter. #XPMY will read all these extra parameters and will write them onto its output file in the beginning of the unused space. On completion of the create run, #XPMY will then pass information across to the compiler so that the latter can read its parameters from disc and not use a paper tape or card reader.

**Notes**

1. If a #CHAIN parameter is present the initial control parameters must include a compiler parameter (see page 5) specifying #XPLX or #XPLZ as the program to be loaded into core when #XPMY deletes itself.
2. The parameters for #XPLX or #XPLZ must be terminated by a PEND parameter, and this then acts as the input file terminator for #XPMY. It therefore has the same effect as does the #STOP parameter in a run without automatic compilation.

**Programs and Segments from Magnetic Tape**

Programs or segments from magnetic tape may be input at any stage under the control of parameters, described below, which may be inserted on paper tape or cards in appropriate positions in the main input file.

Input magnetic tapes must be in subfile format. The program will take segments from subfiles named in the parameters, provided that these subfiles are of suitable types. Output magnetic tapes produced by the PLAN 3 compilers and consolidators and by the magnetic tape COSY editors, and the magnetic tape dumps produced by #XPMZ or #XPMO, are all suitable for input to #XPMY. If it is desired to input subfile format magnetic tapes from other sources, please refer to the section 'Additional Information' at the end of this chapter for a discussion of the format requirements.

The parameters controlling the input of segments from magnetic tape are as follows:

```
#MFILE SUBFILENAME/SEGMENTNAME,N
IN(FILENAME(FGN/RSN),TSN)
```

optionally followed by

```
#CONTINUE
```

or
#REWIND

or

#CLOSE

where

SUBFILENAME = the name of the subfile to be searched for and opened on the magnetic tape specified by the following IN parameter. It may be up to 12 characters long; if it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters. Internal spaces will not be removed and will be included in the count of 12 characters. SUBFILENAME may be omitted, in which case the whole of the operand should be left blank. For the effect of a blank operand see below.

SEGMENTNAME = the name of the first segment to be copied from the specified subfile under the control of this parameter. It may be up to 11 characters long; if it is punched with fewer than 11 characters the program will supply spaces at the right-hand end to complete the 11 characters. In the case of the segment name, the program will remove any internal spaces, and disregard them in the count of 11 characters (this is to conform with compiler practice). SEGMENTNAME may be omitted, in which case the preceding solidus and the \N must also be omitted. If it is omitted, the whole of the subfile will be copied.

N = the number of segments to be copied, starting with the one specified by SEGMENTNAME; in decimal. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted (and SEGMENTNAME is present) only one segment will be copied.

FILENAME

FGN  

RSN  

TSN

} 

FILEAME, together specify the input magnetic tape to be opened and searched for the subfile specified by the preceding #MFILE parameter. Their significance and the rules governing their use are the same as in the magnetic tape COSY system; see page 21 of Chapter 10, under the IN parameter.

If the #MFILE parameter is used with a blank operand, the editor program will search the tape specified by the IN parameter for a subfile with the name specified by the previous #IDENTITY parameter, and copy the whole of that subfile.

#CONTINUE or #REWIND should be used if further segments are to be copied from the same magnetic tape, with no other magnetic tapes being opened in the meantime. If #CONTINUE or #REWIND is used, the IN parameter should be omitted after the next #MFILE.

#CONTINUE should be used if the next segment required occurs later on the magnetic tape. The tape is not rewound; except that if the program reads right down the tape without finding the subfile specified by the #MFILE parameter, then the tape is rewound to beginning of tape, but is left open.

#REWIND should be used if the next segment required occurs earlier on the magnetic tape. The tape is rewound to beginning of tape, but is left open.

#CLOSE should be used if further segments are to be copied from this magnetic tape after obtaining segments from other magnetic tapes. The tape is closed, but the deck is left on line, so that the program can open the tape again when it is next required.

If none of these three optional parameters is used, the program unloads the tape after the required segments have been copied.
Example of an Input File

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC.</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>INI.T.E.R</td>
<td>NAME</td>
<td>SOURCE</td>
<td>(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H.IST.E.R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M.RO.GR.A.B.L.E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#P.RO.GR.A.B.L.E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC.</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.E.N.D.</td>
<td></td>
<td>CONSOLIDATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.EMI.STE.R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Segments of semi-compiled program

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC.</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#S.T.E.E.R</td>
<td></td>
<td>OBJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#M.EL</td>
<td></td>
<td>FRESHSOURCE / SEG4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN(SOURCE TAPE (2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#S.EMI.STE.R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Segments of semi-compiled program

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC.</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#F. I.N.I.S.H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#I.D.E.N.T.I.T.Y</td>
<td></td>
<td>BLOSUBFILE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#S.T.E.E.R</td>
<td></td>
<td>OBJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Segments of source and semi-compiled program.

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC.</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#F. I.N.I.S.H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#S.T.E.P.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Line Printer Output

The parameters are listed on the line printer as they are read. Any parameter lines which are in error are flagged with two asterisks in the left-hand margin. Any parameters or data lines which contain a paper tape format error are printed flagged with *P* in the left-hand margin. If #VARIABLE is present, a list of any source lines that are out of sequence will also be printed flagged with *P*, as explained earlier. As the writing of each subfile is completed, a summary listing of the segments it contains is printed. An example is given below.

After a page heading 'Created by #XPMY', the mark number and the date, each segment is summarized in four columns:

1. The segment name.
2. 'PLAN' if the segment is in PLAN source form, or 'S/C' if it is in semi-compiled form.
3. Against each PLAN source segment only, a number which will always be 0 for #XPMY, except in a file recreate run. The significance of this column for #XPMZ, and for #XPMY in a file recreate run, is explained on page 30.
4. Against each PLAN source segment, the steering information for the segment; against each semi-compiled segment, the #S.EMI.STE.R operand applicable to the segment.

On the first segment's line only, a preceding column shows the name of the composite subfile which contains the program.
At the end of the summary is printed a note of the file name, the file generation number, the version number, the retention period and the cartridge serial number or unit serial number. This is followed by a note of the number of buckets used for the subfile.

The listing relative to the preceding example would appear as follows:

```
CREATED BY #XPMY    #03   11/03/69
FREDSUBFILE   SEGONE   PLAN  0   LIST,OBJECT
SEG2        S/C      CONSOLIDATE
SEG3        S/C      CONSOLIDATE
SEG4        PLAN  0   OBJECT
SEG5        PLAN  0   OBJECT
SEG6        S/C
SEG7        S/C
SOURCEFILE  FGN=3   VN=0   RET=0   SERIAL NO.=777213
NO. OF BUCKETS USED=275
```

After the summary listing of the last subfile created, #XPMY prints a final summary of the file in the following form:

```
CREATED BY #XPMY    #03   11/03/69
UNAMENDED PROGRAMS:
SUBFILEONE
SECONDSUBF
SOURCEFILE  FGN=3   VN=0   RET=0   SERIAL NO.=777213
NO. OF BUCKETS ALLOCATED TO FILE = 560
NO. OF SPARE BUCKETS = 57
EXTENDED BY 80 BUCKETS
```

These lines are printed only in an APPE or APPF run.

The last line is printed only if the file was extended in the course of the run. This extension occurs on the storage unit that holds the end of file area (which is not necessarily the same storage unit as that which holds the file description). The figure printed for the number of buckets allocated shows the size of the file including any extensions that may have been made.

**Line Printer Error Messages**

The following error messages may be output on the line printer:

- **APPEND FILE IS IN WRONG FORMAT**
  
  The output file specified by an APPE or APPF parameter is not in the correct format. The run is abandoned with the console message 'HALTED: - XX'.

- **DUPLICATE subfilename REJECTED**
  
  The subfile name specified by a #IDENTITY parameter is already in use in the file. The input associated with the parameter is ignored and the program proceeds to the next #IDENTITY parameter.

- **SUBFILE name NOT FOUND**

- **SEGMENT name NOT FOUND**

- **NO INPUT FILE OPEN FOR subfilename**

- **FILENAME IS IN WRONG FORMAT**

- **n SEGMENTS NOT FOUND**
  
  A number of segments has been specified in a #MFILE parameter, but the end of the subfile has been reached on the magnetic tape before the specified number of segments has been copied. The number n shows the shortfall in the number of segments. The program proceeds to the next item in the paper tape/card input file.
NO END SUPPLIED, SEGMENT GIVEN ONE
A STEER, a SEMISTEER, a VARIABLE, a FILE, a TAG or a STOP parameter has been read, but no END for the current segment has been detected. The program infers a END for the current segment, and proceeds.

NO MORE SUBFILES CAN BE ACCEPTED
There are already 33 programs in the output file, and another IDENTITY parameter has been read. The program continues reading the basic peripheral input but takes no action on it until the STOP parameter is read, when the run ends normally.

TOO MANY TAG DUMPS INPUT
This message will occur only where amendments subfiles produced by XPMZ form part of the input; see the section ‘Re-creating a DISC COSY File’.

More than 82 amendments subfiles have been input. Any more after this point will be rejected.

Operating Instructions for XPMY
1 If exchangeable disc files are used, load the cartridge or cartridges containing:
   (a) the program XPMY.
   (b) the required compiler, if the compiler parameter is present.
   (c) the file to be used for output.
   (d) the file to be used as a work file, if required.
2 Load any magnetic tapes which are to be used for input.
3 Input the message:
   FI XPMY NAME
   where NAME represents the last four characters of the name of the file containing XPMY.
4 Load the parameters and data in the paper tape reader and/or card reader.
5 If it is required that XPMY should not output a display message to the console typewriter for each subfile it creates, input the console message:
   ON XPMY 20
6 If the initial control parameters are on paper tape, activate by:
   GO XPMY 20
   If the initial control parameters are on cards, activate by:
   GO XPMY 21
7 The program allocates the appropriate reader and the line printer, opens and closes or unloads its disc and magnetic tape files as required, and writes the output file.
8 At the end of the creation of each subfile, XPMY outputs one of the following console messages:
   (a) If the subfile was created successfully and no parameter errors were detected:
      0#XPMY; DISPLAY:-- SERV OK SUBFILENAME
   (b) If any parameter errors were detected, or if any line printer error messages were output for any reason:
      0#XPMY; DISPLAY:-- SERV ER SUBFILENAME
   These console display messages may, if required, be inhibited by setting a switch before entering XPMY (see operating instruction number 5, above).
9 When the run is completed the program deletes itself, with a message in the form:
   (a) if no compiler parameter was present:
      0#XPMY; DELETED:-- MY
(b) if a compiler parameter was present:

00XPMY; DELETED= F1 #ABCD #NAME N

where #ABCD, #NAME and N are as specified in the compiler parameter.

EXCEPTION CONDITIONS

The following messages may be output on the console typewriter during the running of the program:

1 00XPMY; HALTED= INCORRECT FILE PARAMETERS. FIX & RESTART

This message is output if there is a format error in any of the initial control parameters, or if RENF is used in conjunction with OUTE or APPE, or if RENE is used in conjunction with OUTF or APPF. The parameters should be corrected and the program restarted from step 4.

2 00XPMY; HALTED= NEEDS TAPE NNNN or
   00XPMY; HALTED= NEEDS DISC NNNN

These messages are output if a tape serial number or a cartridge serial number respectively has been checked and found incorrect. The correct tape or cartridge as specified in the appropriate parameter should be loaded, and the program resumed by:

   GO #XPMY

3 00XPMY; HALTED= MASTER FILE NOT FOUND

This message is output if the file specified by the OUTE or OUTF or APPE or APPF parameter cannot be found. Either the correct cartridge should be loaded and the program resumed by:

   GO #XPMY

or the parameter should be corrected and the program restarted by:

   GO #XPMY 20  or  GO #XPMY 21

4 00XPMY; HALTED= WORK FILE NOT FOUND

This message is output if the permanent file specified by the OUTW parameter cannot be found. Either the correct cartridge should be loaded and the program resumed by:

   GO #XPMY

or the parameter should be corrected and the program restarted by:

   GO #XPMY 20  or  GO #XPMY 21

Alternatively, the console message:

   GO #XPMY 29

will cause a scratch file to be opened as the work file, on any cartridge or unit which is on line.

5 00XPMY; DISPLAY= NEEDS DISC NNNN

This message is output immediately before either of messages 3 or 4 above, if the parameter specifying the file to be opened included a CSN field.

6 00XPMY; HALTED= INSUFFICIENT BLOCKS FOR WORK FILE

This message is output if there is insufficient room for the work file which may be:

(a) on the cartridge or unit containing the output file, if no OUTW parameter was present; or
(b) in the permanent file specified by the OUTW parameter; or
(c) on the cartridge or unit specified by the OUTW parameter where this specifies a scratch file.

The console message:

   GO #XPMY 29

will enable the restrictions imposed by the presence or absence of an OUTW parameter to be disregarded, and Executive to search for a scratch file of sufficient size on any cartridge or unit which is on line. If the same
message then occurs, it may be possible to make more space available by deleting another program which is using disc scratch files, or, if exchangeable discs are used and more than one transport is available, by loading another cartridge which has more spare blocks. The message:

GO \#XPMY 29

would then be repeated. If still unsuccessful, the run must be abandoned.

7 0\#XPMY; LOAD FILENAME

where FILENAME is the name of a magnetic tape input file specified by an IN parameter associated with a #MFILE parameter. Make the tape available and resume by:

GO \#XPMY

If the tape cannot be made available, the console message:

GO \#XPMY 28

will cause the program to continue the run, omitting any segments that were to have been copied from that tape.

8 0\#XPMY; HALTED:-- TAPE MISREAD

This message is output if a misread is detected while the program is copying a segment onto the output file from a magnetic tape file. The message:

GO \#XPMY

will cause the program to continue to run, omitting the segment in which the misread occurred.

9 0\#XPMY; HALTED:-- NO SPACE AVAILABLE ON *NNNNNN
0\#XPMY; HALTED:-- SYSTEM CONTROL AREA FULL ON *NNNNNN

Either of these messages may occur when the program is seeking to extend the output file. If it is possible to make more space available on the cartridge or unit by deleting another program which is using disc scratch files, then after doing so the message:

GO \#XPMY

will cause the program to make another attempt to extend the output file. Alternatively if more than one transport or unit is available, the message:

AL \#XPMY 7 *NNNNNN

(where NNNNNNN represents the octal serial number of an alternative cartridge or unit) followed by:

GO \#XPMY 27

will cause the output file to be extended onto the second cartridge or unit, if space is available. If space on the second cartridge or unit is not available, the appropriate HALTED message will again occur.

10 0\#XPMY; HALTED:-- XX

This message is output if the output file specified by an APPE or APPF parameter is not in the correct format. Abandon the run.

11 0\#XPMY; DISPLAY:-- PL

This message is output if a paper low condition has been detected on the line printer. The line printer is disengaged. The paper should be replenished and the line printer reallocated to continue the run.

UPDATING A DISC COSY FILE

Two programs are available for updating a DISC COSY file: #XPMZ which will run only on processors with at least 16K words of core store, and #XPMX for use on processors with 8K words of core store. Both programs as supplied have a priority of 80.

The file being updated must be in standard subfile format as produced by #XPMX, #XPMY or #XPMZ.

An update run using either program may comprise any combination of the following:

1 The insertion, deletion or replacement of complete programs, in PLAN source form, the new programs being input from paper tape or cards
2. The insertion, deletion or replacement of complete segments, in PLAN source form, the new segments being input from paper tape or cards.

3. The insertion, deletion or replacement of lines within PLAN source segments.

4. The provision of new steering information for a subsequent compilation run.

In addition to the above, the following facilities are available in #XPMZ only:

1. Semi-compiled segments may be inserted, deleted or replaced.

2. New programs or segments to be incorporated into a file may be input from magnetic tape files or other disc files, provided that these are also in subfile format.

3. Two extra types of amendment are available as follows:

   (a) The insertion, deletion or replacement of only the comment fields of lines within PLAN source segments.

   (b) The deletion of all amendments made to a source segment since the file was last reorganised. (Note: "reorganised" in this context means updated by a copy-and-amend run. See under "Editing Modes of #XPMX and #XPMZ", below).

4. #XPMZ can in addition provide dumps of segments or of complete programs onto magnetic tape and/or paper tape or cards, and/or listings of segments or programs on the line printer. Alternatively, dumps of all amendments made to particular programs or individual segments since the file was last reorganised by a copy-and-amend run can be provided.

#XPMX and #XPMZ are fully compatible in their file formats, except that a file to be input to #XPMX must not contain any semi-compiled segments.

**Editing Modes of #XPMX and #XPMZ**

The Editor programs may update a file in a choice of two ways, known respectively as amend-in-situ and copy-and-amend.

Amend-in-situ creates the updated file by writing the amendments within the area occupied by the input file, utilizing the spare space at the end of the file. New and replacement segments are inserted as new subfiles with their associated subfile descriptions; deleted and replaced segments are not erased, but have their subfile descriptions removed, so that they may no longer be accessed. Line amendments within a source segment are made by creating or updating an amendments subfile, consisting of tags pointing to the original source subfile together with the new source lines. Each updating run using this mode increases the version number of the output file by one, unless otherwise specified by a RENE or RENF parameter.

Copy-and-amend uses a new file area for the updated file. The file is reorganized to eliminate all deleted and replaced segments, and to incorporate the amendments subfiles into their associated source subfiles. The input file remains in its original area. The version number of the reorganized file is set to zero, unless otherwise specified by a RENE or RENF parameter; except that if that parameter is omitted, and no FGN2 field is specified for the OUTE or OUTF parameter (see page 18) then the version number is increased by one.

The foregoing is all that the user needs to know about the editing modes of #XPMX and #XPMZ, but, for those who are interested, further details may be found in the section headed 'Additional Information', at the end of this chapter.

**Hardware Requirements for #XPMX and #XPMZ**

The hardware requirements for #XPMX are:

- 5888 words of core store.
- One paper tape reader and/or one card reader.
- One line printer.

and either:

- Two or more exchangeable disc stores.

or:

- One or more fixed disc stores.
The hardware requirements for #XPMZ are:

- 9K words of core store.
- One paper tape reader and/or one card reader.
- One line printer.
- One paper tape punch and/or one card punch (optional).
- One or more magnetic tape decks (optional).

and either:

- Two exchangeable disc stores.
- One or more further exchangeable disc stores (optional).

or:
- One or more fixed disc stores.

Both programs always require a minimum of either three or four disc files: the file from which the program is to be loaded, the file to be updated, a work file and, if the copy-and-amend editing mode is being used, the file to be used for output. In #XPMZ only, optional additional disc input files may be specified.

With the exception of the file from which the program is to be loaded, all disc files used during a run must be on the same type of disc, exchangeable disc or fixed disc. If exchangeable discs are used a minimum of two drives is required, but the program will run more efficiently if each file is on a different cartridge. If fixed discs are used, the files may be on the same unit or on different units.

The file containing the program is closed as soon as the loading process has been completed. The other files are specified by parameters, as described under 'Input Data' below. All disc files used for input must be in the format produced by the DISC COSY editor programs.

All magnetic tape input files must be in the format specified for #XPMY (see page 8).

Basic Peripheral Usage, #XPMX and #XPMZ

A paper tape reader or a card reader is required to read in control parameters and line amendments. New programs or segments may also be read from paper tape or cards. The required peripheral, a paper tape reader or card reader, is allocated initially as determined by the choice of entry point (see 'Operating Instructions' on page 33) and is released when either all the input data has been read or there is a change of input medium. The input medium can be changed at any time by a #SWITCH directive. When this is encountered in the input data the current reader is released and one of the other type is allocated.

A line printer is allocated as LPO when the program is entered, and is released at the end of the run.

Magnetic Device Usage, #XPMX

OBLIGATORY DISC FILES

The input file is opened as ED4.

The work file is opened as ED1 when running in amend-in-situ mode, and as ED3 when running in copy and amend mode.

OPTIONAL DISC FILE

The output file, used when running in copy and amend mode, is opened as ED1.

Magnetic Device Usage, #XPMZ

The medium used, exchangeable disc or fixed disc, is specified by the parameters, as described under 'Input Data' below.

OBLIGATORY DISC FILES

The input file is opened as ED4 or FD4.

The work file is opened as ED1 or FD1 when running in amend-in-situ mode, and as ED3 or FD3 when running in copy-and-amend mode.

OPTIONAL DISC FILES

The output file, used when running in copy-and-amend mode, is opened as ED1 or FD1.

If programs or segments are to be copied from a disc file, the file is opened as ED2 or FD2.
OPTIONAL MAGNETIC TAPES

If a magnetic tape is to be opened to receive a dump of segments on the updated file, the file is opened as MT0.

If programs or segments are to be copied from a magnetic tape, the tape is opened as MT1.

Input Data
The input data consists of:

1. Parameters and amendments (including new programs or segments) to be read from paper tape and/or cards.
2. A disc file in the format produced by the DISC COSY editors, to be updated.
3. Optionally, for #XPMZ only, other disc files in the format produced by the DISC COSY editors, from which programs or segments are to be copied onto the updated file.
4. Optionally, for #XPMZ only, magnetic tape files in the format specified for #XPMY, from which programs or segments are to be copied onto the updated file.

Of these, the amendments file, to be read from paper tape and/or cards, may be regarded as consisting of three parts:

1. Initial control parameters.
2. A batch of amendments, the amendments for each program or segment being preceded by their own set of control parameters, and optionally followed by #FINISH. Control parameters for amendments to be input from other disk files or from magnetic tape files will also appear here.
3. A batch terminator, and optionally, parameters for automatic compilation.

Each of these three parts will now be considered in turn.

Both for the initial control parameters and for the individual amendment parameters, each parameter line starts at the first character position of a paper tape line or at column 1 of a card. Paper tape lines are terminated by a newline character and have a maximum permitted size of 128 internal characters; beta shift characters or delta shift characters other than $, ], ↑, =, newline, or horizontal tabulation are not accepted. Paper tape lines are converted to card image format; after conversion they must not contain more than 80 characters. Any paper tape lines which contravene any of these rules are flagged with a preceding *P* in the line printer output.

Initial Control Parameters
The initial control parameters comprise one compulsory and further optional parameters. They define the main disc input file; optionally, other disc files and/or magnetic tapes to be opened for input or output; and the compiler to be used, if the output is to be compiled immediately. These parameters may appear in any order, provided that they appear before the first #IDENTITY parameter (described under 'Amendments and Control Parameters' below).

The initial control parameters are as follows. They conform to the same generalized format as the parameters for PLAN 3 compilers discussed on page 21 of Chapter 7. The first three described have alternative forms, depending on whether the files are on exchangeable disc or on fixed disc.

1. INE or INF

This parameter is compulsory. INE defines the exchangeable disc file to be opened as the input file to be updated; INF defines the fixed disc file to be opened as the input file to be updated.

The parameter has the following format:

\[ \text{INE(FILENAME(FGN),CSN)} \] or \[ \text{INF(FILENAME(FGN))} \]

where

- **FILENAME** = the existing 12-character file name. If the file name is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.
- **FGN** = the file generation number, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, the associated brackets should also be omitted. If it is omitted, or if it is expressed as \(-1\), then the highest-numbered generation of the specified file on line will be opened.
- **CSN** = the serial number, in octal, in the range octal 0 to octal 777777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.
2 OUTE or UTF

This parameter is optional. If present, it causes the editor program to work in copy-and-amend mode. OUTE defines the exchangeable disc file which is to be opened to receive the output; UTF defines the fixed disc file which is to be opened to receive the output.

If this parameter is absent, the editor program works in amend-in-situ mode.

The parameter has the following format:

\[
\text{OUTE}(/\text{FILENAME}(\text{FGN}1=\text{FGN}2),\text{CSN}) \quad \text{or} \quad \text{UTF}(/\text{FILENAME}(\text{FGN}1=\text{FGN}2))
\]

where

\[
\text{FILENAME} = \text{the existing 12-character file name of a permanent file which is to be opened and overwritten by the editor program's output. If the file name is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.}
\]

\[
\text{FGN}1 = \text{the file generation number currently existing on the file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, or if it is expressed as -1, then the highest-numbered generation of the specified file on line will be opened.}
\]

\[
\text{FGN}2 = \text{the new file generation number to be written to the output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding = sign should also be omitted. If it is omitted the file generation number will be left unchanged.}
\]

\[
\text{CSN} = \text{the serial number, in octal, in the range octal 0 to octal 77777, of the cartridge containing the file to be opened. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the cartridge serial number will not be checked.}
\]

If FGN1 and FGN2 are both omitted the associated brackets are also omitted; thus:

\[
\text{OUTE}(/\text{FILENAME},\text{CSN}) \quad \text{or} \quad \text{UTF}(/\text{FILENAME})
\]

The version number of the file will be set to zero if FGN2 is present, or will be increased by one if FGN1 is omitted.

3 RENE OR RENF

This parameter is optional. It causes the output file to be renamed. In copy-and-amend mode, the file specified by the OUTE or UTF parameter is renamed; in amend-in-situ mode the file specified by the INE or INF parameter is renamed. A RENE parameter must not be used in conjunction with an INF or an UTF parameter, nor must a RENF parameter be used in combination with an INE or OUTE parameter.

The parameter has the following format:

\[
\text{RENE}(/\text{FILENAME}(\text{FGN}/\text{VN})) \quad \text{or} \quad \text{RENF}(/\text{FILENAME}(\text{FGN}/\text{VN}))
\]

where

\[
\text{FILENAME} = \text{the new file name which is to overwrite the existing file name on the output file. It may be up to 12 characters long. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.}
\]

\[
\text{FGN} = \text{the file generation number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted. If it is omitted, the output file will be given a generation number of zero. FGN must not be expressed as -1.}
\]

\[
\text{VN} = \text{the version number to be written to the renamed output file, in decimal, in the range 0 to 4095. It may be omitted, in which case the solidus should also be omitted. If it is omitted, the output file will be given a version number of zero.}
\]

If FGN is omitted, and VN is present, the solidus must also be present; thus:

\[
\text{RENE}(/\text{FILENAME}(\text{/VN})) \quad \text{or} \quad \text{RENF}(/\text{FILENAME}(\text{/VN}))
\]
If FGN and VN are both omitted the associated brackets are also omitted; thus:

\[ \text{RENE(FILENAME) or RENF(FILENAME)} \]

If a RENE or RENF parameter is present, the file generation number and version number to be written to the output file is determined solely by the RENE or RENF parameter; if, when RENE or RENF is present, an OUTE or OUTF parameter specifies FGN2, that field of the OUTE or OUTF parameter will be overridden.

4 OUTW (#XPMZ only)

This parameter is optional. It defines the disc file to be opened as the work file. In a #XPMX update run, or in a #XPMZ update run where this parameter is omitted, a scratch file will be opened on the same cartridge or unit as the input file.

If it is desired to open a permanent file as the work file, the parameter has the following format:

\[ \text{OUTW(FILENAME(FGN1=FGN2),CSN)} \]

where \( \text{FILENAME} \) = the existing 12-character file name of a permanent file which is to be opened and overwritten by the editor program's work file. If the file name is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

The significances of FGN1, FGN2 and CSN, and the rules governing their use, are as described under OUTE or OUTF above. CSN is not applicable to fixed discs, and should always be omitted, together with its preceding comma, when a file on that medium is being specified.

If it is desired to open a scratch file as the work file, but on a different cartridge or unit from the output file, then the file name is expressed as #####, and the cartridge serial number or unit serial number is quoted; thus:

\[ \text{OUTW(#####, CSN) or OUTW(#####, USN)} \]

where \( \text{USN} \) = the unit serial number, in octal, in the range octal 1 to octal 777777, of the fixed disc unit containing the file to be opened. It may be punched with or without a preceding # or * sign.

5 OUT or APP (#XPMZ only)

This parameter is applicable only to the copy-and-amend mode, and is optional in that mode. It defines a magnetic tape which is to be opened to receive a dump of some or all of the segments on the updated file, as determined by #COPYSEG or #COPYPROG parameters (see under 'Amendments and Control Parameters' below).

The parameter has the following format:

\[ \text{OUT(FILENAME(FGN/RSN),TSN)} \]

or

\[ \text{APP(FILENAME(FGN/RSN),TSN)} \]

where \( \text{FILENAME} \) = the existing 12-character file name. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

\( \text{FGN} \) = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted or expressed as zero the file generation number on the tape is not checked.

\( \text{RSN} \) = the reel sequence number, in decimal, in the range 0 to 511. The reel sequence number will always be checked, but if it is zero it may be omitted from the parameter, in which case the preceding solidus should also be omitted.

\( \text{TSN} \) = the tape serial number, in octal, in the range octal 0 to octal 37777777. It may be punched with or without a preceding # or * sign. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted or expressed as zero the tape serial number will not be checked.
If FGN is omitted, and RSN is present, the solidus must also be present; thus:

\[ \text{OUT}(	ext{FILENAME}/	ext{RSN}, 	ext{TSN}) \text{ or } \text{APP}(	ext{FILENAME}/	ext{RSN}, 	ext{TSN}) \]

If FGN and RSN are both omitted, the related brackets are also omitted; thus:

\[ \text{OUT}(	ext{FILENAME}, 	ext{TSN}) \text{ or } \text{APP}(	ext{FILENAME}, 	ext{TSN}) \]

The difference in effect of the two forms of the parameter is as follows. OUT causes the magnetic tape file to be positioned immediately after the header label, so that the dumped segments overwrite the previous contents of the tape. APP causes the magnetic tape file to be positioned immediately before the trailer label, so that the dumped segments are appended onto the previous contents of the tape.

If a tape labelled SCRATCH TAPE is required to be opened as the dump tape, then it should be declared in the OUT parameter with SCRATCH TAPE as its file name; if a specific tape labelled SCRATCH TAPE is required to be opened, then the tape serial number should additionally be quoted.

If it is intended to rely on Executive's search for a tape with an expired retention period to locate a dump tape, then it will be necessary to use an OUT parameter with FILENAME written as ####, thus:

\[ \text{OUT}(	ext{####}, 	ext{TSN}) \]

If it is required to pick up a tape of a given serial number and an expired retention period as the dump tape, and there is any uncertainty as to the existing file name, then it is permissible to use a parameter of the form:

\[ \text{OUT}(	ext{####}, 	ext{TSN}) \]

The sequence of events in this case is that Executive locates the lowest numbered deck containing a tape with an expired retention period, opens the tape, and relabels it SCRATCH TAPE. #XPMZ then proceeds to check the tape serial number. Should this not be the required one, the program unloads the tape and halts with the message 'HALTED:-- NEEDS TAPE NNNN'.

This gives the operator the opportunity to put up the correct tape, or to put lower numbered decks off line, as may be necessary, before restarting by the console message 'GO #XPMZ'. Any tapes opened and subsequently closed in this way have of course had their original header labels overwitten; the facility should therefore be used with care.

If an OUT or APP parameter is present when #XPMZ is working in amend-in-situ mode (that is, when no OUTE parameter is present) then the parameter is accepted but ignored.

6 REN (#XPMZ only)

This parameter is applicable only to the copy-and-amend mode, and is optional in that mode. It may optionally be present only if an OUT parameter is also present. It causes the tape specified by the OUT parameter to be renamed.

The parameter has the following format:

\[ \text{RE}(	ext{FILENAME}(\text{FGN}/	ext{RSN}, 	ext{RET})) \]

where FILENAME = the 12-character file name to be written to the tape. If this is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

FGN = the file generation number, in decimal, in the range 0 to 8388607. It may be omitted. If it is omitted the tape is given a file generation number of zero.

RSN = the reel sequence number, in decimal, in the range 0 to 511. It may be omitted, in which case the preceding solidus should also be omitted. If it is omitted, the tape is given a reel sequence number of zero.

RET = the number of days retention period to be given to the tape, in decimal, in the range 0 to 4095. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted the tape is given a retention period of 4095.

If FGN is omitted, and RSN is present, the solidus must also be present; thus:

\[ \text{RE}(	ext{FILENAME}/	ext{RSN}, 	ext{RET}) \]

If FGN and RSN are both omitted, and RET is present, the preceding comma must also be present; thus:

\[ \text{RE}(	ext{FILENAME}, 	ext{RET}) \]
If FGN, RSN and RET are all omitted, the related brackets are also omitted; thus:

RUN(FILLENAME)

7 The compiler parameter

This parameter is optional. It causes a specified program (normally a compiler) to be brought into core when the editor program deletes itself. If the compiler parameter is present, then on the completion of its run the editor program deletes itself with a DELTY 'FIND' message. The editor program provides the 'F1' portion of this message; the variable portion is obtained from the compiler parameter. The compiler parameter may therefore contain any of the elements appropriate to a DELTY 'FIND' message in the particular environment in which it is being used; thus, in suitable environments, peripheral requests may be included in the parameter, in addition to the fields discussed below.

The generalized format of the compiler parameter is:

NEXT(Variable portion of DELTY 'FIND' message)

This may usually be formalized as:

NEXT(#ABCD #NAME N)

where

#ABCD = the name of the program to be brought into core when the editor program deletes itself.

NAME = the last four characters of the name of the file containing #ABCD. The name of this file must be of the form PROGRAM#NAME. #NAME may be omitted where NAME is the same as ABCD.

N = the amount of core into which the program is to be loaded; in decimal, or in octal with a preceding asterisk. It may be omitted. If it is present, it will override the core request contained in the program's request block.

The spacing between these fields of the variable portion of this parameter is immaterial, provided that no spaces occur within the fields and that the whole variable portion does not exceed 37 character positions. For further details of what may be permitted in a FIND message, see the appropriate console operating manual.

In #XPMZ only, if the program specified by this parameter is a PLAN compiler having a name #XPL-, then NEXT and the brackets may be omitted, together with the first hashmark, so that the parameter becomes:

XPL- #NAME N

where #NAME and N are as defined above. This form of the compiler parameter will not be accepted by #XPMX.

Amendments and Control Parameters

The initial control parameters as described above are followed by a batch of amendments, each with its own preceding control parameters, and each optionally followed by #FINISH. Control parameters for segments to be brought into the updated file from other disc files or from magnetic tape will also appear here.

The parameter lines have a format similar to PLAN compiler directives. Where a parameter has an operand, the operand must be punched starting from character 16 of a paper tape line or column 16 of a card. If horizontal tabulation characters are used in paper tape input they will be expanded to the field format of PLAN source cards.

Segments may be amended in any order, irrespective of their sequence on the file.

The control parameters are as follows:

1   #IDENTITY

This parameter is compulsory. It must be the first parameter for each amendment. It identifies the program or segment which is to be amended; or, in the case of a program new to the file, it establishes the name of the new subfile which is to be created to contain that program.
The parameter has the following format:

```
#IDENTITY SUBFILENAME/SEGMENTNAME,N
```

where `SUBFILENAME` = the name of the subroutine containing the program to which the amendment relates, or, if the amendment is a program new to the file, the name to be given to the new subroutine into which the program is to be put. It may be up to 12 characters long. If it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters.

`SEGMENTNAME` = the name of the segment to which the amendment relates. It is necessary only if the amendment relates to a specific segment; if the amendment relates to a whole program, then `SEGMENTNAME` is omitted, in which case the preceding `/` should also be omitted. If the amendment comprises the insertion of a new segment, `SEGMENTNAME` should be the name of the segment after which the new one is to be inserted; except that if the new segment is to be inserted as the first segment in the subroutine, `/SEGMENTNAME` is omitted.

`SEGMENTNAME` may be up to 11 characters long; if it is punched with fewer than 11 characters the program will supply spaces at the right-hand end to complete the 11 characters. In the case of the segment name, the program will remove any internal spaces, and disregard them in the count of 11 characters (this is to conform with compiler practice).

`N` = a number indicating which version of the segment specified is to be amended (or, in the case of a new segment, after which version of the specified segment the new segment is to be inserted). If the subroutine contains more than one segment with the same name, or if the segment exists in the subroutine in both PLAN source and semi-compiled forms, the number `N` following the segment name indicates that it is the `N`th version of that segment which is being specified. `N` may be omitted, together with its preceding comma, where the first or only version of the named segment is being specified.

Note that if a prior amendment in the same run deletes or inserts another segment of the same name, and the deleted or inserted segment comes before the `N`th version in the subroutine, then the `N`th version becomes the `(N-1)`th or `(N+1)`th version respectively. The value of `N` should be specified with this fact in mind.

The program will run most efficiently if the order in which the segments are amended is the reverse of that in which they appear in the subroutine. A segment should not be specified in more than one amendment, except that it may be amended by a `#LINEDIT` amendment and referenced by a `#NEWSEG` amendment (see the amendment type parameter below). Programs may be amended in any order, but all amendments to one program should be together in the paper tape/card input file.

The remaining parameters are all optional and may be input in any sequence.

2 `#STEER`

This parameter is optional, as it is not always required to compile the program after a file updating run. It specifies the steering information which is to be inserted into all unamended source segments in the specified subroutine and also into any amended source segments for which no `#SEGSTEER` parameter is input, subject to the rules discussed below. When the program is subsequently submitted to the compiler only those segments with steering information present will be compiled.

The parameter has the following format:

```
#STEER Steering information
```

where `Steering information` = any valid steering line for the compiler which is to be used.

The steering information specified by the `#STEER` parameter will be inserted for:

(a) the segment currently being amended, if this is a source segment. If the amendment is an addition of a new segment, the steering information will be inserted for the new segment provided that it is in PLAN source form; not, at this stage, for the segment specified in the associated `#IDENTITY` line;

(b) the segments affected by subsequent amendments to the same program, provided that they are source segments;
(c) after all the amendments for this program have been read, all unamended source segments of the program; unless overridden for particular segments by a #SEGSTEER parameter (described below), or until a further #STEER is encountered in a later amendment.

#STEER may be used with more than one amendment to a program, each #STEER operand applying in the manner indicated above until the next #STEER is met. The operand of the final #STEER will be applied to all the unamended source segments of the program.

#STEER may be used with a blank operand to denote that no steering information is to be inserted in source segments (except where overridden by a #SEGSTEER parameter) until a further #STEER is met. It is, however, unnecessary to do this unless a previous amendment to the program in this run has included a #STEER with a non-blank operand.

Whether or not #STEER is present, any previously existing steering information is removed from the subfile when it is updated.

Note: The action on steering information for unamended programs depends on the mode in which the editor program is working. When updating in amend-in-situ mode, #XPMZ sets a switch (known as the do-not-compile switch) for all unamended programs in the file, but leaves their steering information unaltered. The effect of this is that if the file is subsequently submitted to a DISC COSY compiler and the SUB compiler parameter is not used, then only the amended programs for which steering information was specified will be compiled; but if the SUB compiler parameter is used, then any program which has steering information present may be compiled. (The SUB compiler parameter is described in Chapter 7.) When updating in copy-and-amend mode, any steering information for programs not amended in the course of the run is omitted from the output file.

If it is desired to change the steering information present for a program without making any other amendments, or to preserve steering information for an unamended program in a copy-and-amend run, this can be done by means of a dummy amendment consisting solely of the #IDENTITY, #STEER and #FINISH parameters.

3 #SEGSTEER

This parameter is optional, and should be input only with an amendment relating to a PLAN source segment. It specifies the steering information which is to be inserted in the source segment where this differs from the overall steering information as determined by #STEER, or where no overall steering information is provided. This facility is useful when listing or monitoring of amended segments only is required.

The parameter has the following format:

```
#SEGSTEER Steering information
```

where Steering information = any valid steering line for the compiler which is to be used.

#SEGSTEER may be used with a blank operand to denote that no steering information is to be inserted for this segment.

4 #SEMISTEER (#XPMZ only)

This parameter is optional. It specifies whether or not semi-compiled segments are to be copied across to the compiler output file when the editor output file is subsequently submitted to the compiler; and thus whether or not these segments are to be made available to the consolidator. If this parameter is used in a #XPMX run, it will be flagged and ignored.

The parameter has the following formats:

```
#SEMISTEER CONSOLIDATE
```

or

```
#SEMISTEER
```

with the operand field blank.

If none of the semi-compiled segments in the subfile is to be consolidated, the parameter should be omitted. If all are to be consolidated, the CONSOLIDATE form should be input with the first amendment relating to a semi-compiled segment, or with any prior amendment. If only some of the semi-compiled segments are to be consolidated, the parameter may be used with different amendments as many times as necessary, a blank operand indicating that segments relating to subsequent amendments are not to be consolidated, and
a CONSOLIDATE operand indicating that segments relating to subsequent amendments are to be consolidated. Each #SEMISTEER is operative until the next one is met, or until there are no more amendments for this program.

Semi-compiled segments to which no amendments have related will be steered in accordance with the last #SEMISTEER input.

Dummy amendments consisting solely of #IDENTITY, #SEMISTEER and #FINISH parameters can be used if necessary.

5 #DUMPSEG or #DUMPPROG (#XPMZ only)

These parameters are alternatives. They may only be used in amend-in-situ mode, and are optional in that mode. #DUMPSEG causes the amendments subfile of the segment specified in the associated #IDENTITY parameter to be dumped onto paper tape or cards. #DUMPPROG causes all the amendments subfiles relating to the program to be dumped onto paper tape or cards.

The parameters have the following formats:

#DUMPSEG xP
#DUMPPROG xP

where xP = the symbolic name of the required peripheral (TP or CP). If it is desired to dump onto both paper tape and cards simultaneously, then TP and CP can appear together in the operand field, separated by a comma.

#XPMZ will accept the latter parameter in either of the two forms #DUMPPROG or #DUMPPROG.

#DUMPSEG is meaningful only where the amendment type parameter is #LINEDIT, or is not present.

An amendments subfile dump comprises, for each segment:

(a) A header record in the format:

#TAG /SEGMENTNAME,DATE

where SEGMENTNAME is eleven characters, including trailing spaces.

DATE is the four-character representation of one binary word containing the Executive response to a GIVE with N(M) = 0 instruction.

(b) A number of records, each containing either a tag pointing to the lines in the source subfile which are to be compiled before the next insertion lines are read, or a source line to be inserted. Sequencing will be inserted for each record output, whether tag or source line, on cards or paper tape.

(c) A trailer record, in the format:

#FINISH γN

where γ = a space

N = the total number of records in the dump including #TAG and #FINISH, punched as a four-character representation of a binary word.

The operands of both #TAG and #FINISH commence in character position (or column) 16. Horizontal tabulation characters are not used.

When amendments subfiles are dumped, twenty inches of run-out or one blank card is output after each subfile.

Dummy amendments consisting solely of #IDENTITY, #DUMPSEG or #DUMPPROG and #FINISH parameters can be used if necessary.

6 #COPYSEG or #COPYPROG (#XPMZ only)

These parameters are alternatives. They may only be used in copy-and-amend mode, and are optional in that mode. #COPYSEG causes a copy of the segment specified in the associated #IDENTITY parameter (or, where the amendment type parameter is #NEWSEG or #REPSEG, the segment being inserted) to be dumped onto paper tape, cards and/or magnetic tape, and/or to be listed on the line printer. #COPYPROG causes the whole program to be similarly copied and/or listed.
The parameters have the following formats:

#COPYSEG  p1,p2,p3,p4
#COPYPROG  p1,p2,p3,p4

where  p1,p2,p3,p4  =  the symbolic names of the required peripherals: TP,CP,MT or LP. Any one, two, three or four of these may be specified, separated by commas.

The dumping and/or listing takes place after all the amendments relating to the program have been completed, so that the segments are dealt with in the sequence in which they appear in the subfile. This need not necessarily be the sequence in which the amendments for these segments were input from the amendments file.

If TP is specified, the resulting dump onto paper tape does not have any sequencing inserted, nor does any sequencing which may exist on the file appear on the paper tape dump. Complete words of trailing spaces are removed. Horizontal tabulation characters are not used. Twenty inches of run-out is output after each segment.

If CP is specified, sequencing is inserted in the resulting dump onto cards, both for PLAN source and for semi-compiled segments. The sequencing is inserted in tens, i.e., starting with a one in column 79, each segment being sequenced separately. A blank card is output after each segment.

If MT is specified, an OUT or APP parameter must be present among the initial control parameters. The program or the specified segments are dumped into a composite subfile, within which each group of consecutively dumped segments of like kind (PLAN source or semi-compiled, as the case may be) will be put into a separate simple subfile. It follows that the resulting composite subfile will not be of the same format as one produced by the magnetic tape COSY compilation system unless the disc file, or the selection of segments dumped from it, is so organized that all the source segments precede all the semi-compiled segments. The composite subfile is given the same subfile name as the subfile being dumped, that is, the name which appeared in the #IDENTITY parameter.

If LP is specified, the lines printed are sequentially numbered within each segment. For source segments the sequence field is printed so as to show any sequencing which may exist on the file, except that if CP is also specified then the sequencing output on the dump cards is substituted in the listing for any sequencing which may exist on the file. If CP and LP are both specified for semi-compiled segments, then the card sequencing for those segments will also be printed.

If #COPYSEG is used in conjunction with the amendment type parameter #REPSEG, then it is the replacement segment only which is copied, not the segment being replaced. Dummy amendments consisting solely of #IDENTITY, #COPYSEG or #COPYPROG and #FINISH parameters can be used if necessary.

Amendment type

This parameter is optional. It is normally present, except where a dummy amendment is being used to insert steering information, to obtain a dump of the program, segments or amendments subfiles, or to obtain a listing. The amendment type will be one of the following:

Available in #XPMX and #XPMZ:

#NEWPROG, #DELPROG, #REPPROG
#NEWSEG, #DELSSEG, #REPSEG
#LINEDIT

Available in #XPMZ only:

#COMMENT, #DELAMEND

#NEWPROG, #DELPROG and #REPPROG respectively adds, deletes or replaces the complete contents of a program's subfile.

#NEWSEG, #DELSSEG and #REPSEG respectively adds, deletes or replaces one or more segments of the program. #LINEDIT introduces a set of detailed amendments to a segment.

#COMMENT is used when only the comment in a segment needs amending. It is available in copy-and-amend mode only. #DELAMEND causes the amendments subfile of a PLAN source segment to be deleted. Neither of these amendment types is available in #XPMX.
#NEWPROG
This causes a new program to be inserted in the file.
The parameter has the following format:

#NEWPROG V

where V is optional. In #XPMZ, the presence of V causes the two words of card sequencing to be removed from all PLAN source statements controlled by this parameter, before their records are written to the file. #XPMX always removes card sequencing from PLAN source lines, and the presence or absence of V therefore has no effect. The effect is that reading and writing the file will be speeded up: see the discussion of #VARIABLE on page 6 of this chapter.

When #XPMX or #XPMZ is removing card sequencing it checks first that the sequence is correct, and outputs on the line printer a list of any lines that are out of order, flagging the lines with a preceding *P*.
The records are written to the file, however, in the sequence in which they were input.

In #XPMX, the last of the control parameters for a #NEWPROG amendment is followed by the new segments in PLAN source form, on cards and/or paper tape.

In #XPMZ, the last of the control parameters for a #NEWPROG amendment is followed by any combination of the following:

(a) PLAN source segments or cards and/or paper tape;
(b) Semi-compiled segments on cards and/or paper tape;
(c) Parameters specifying PLAN source and/or semi-compiled segments to be brought in from magnetic tape or from another disc file; see under the appropriate sub-headings below.

Any segment of the new program may appear in either or both of PLAN source or semi-compiled forms, and different versions of a segment having the same name may be input if desired.

#DELPROG
This causes a program to be deleted from the file.
The parameter has the following format:

#DELPROG

with the operand field blank.

#REPPROG
This causes a program to be deleted from the file and replaced by a new one.
The parameter has the following format:

#REPPROG V

where V is optional, and is as described under #NEWPROG.
The parameter is accepted in either of the forms #REPPROG or #REPROG.
If this parameter is present, the last of the control parameters for this amendment is followed by the replacement segments and/or parameters, as described under #NEWPROG.

#NEWSEG
This causes one or more new segments to be inserted in the file.
The parameter has the following format:

#NEWSEG V

where V is optional, and is as described under #NEWPROG.
If #NEWSEG is present, the last of the control parameters for this amendment is followed by the new segments and/or parameters, as described under #NEWPROG.
The new segment(s) will be inserted in the file so as immediately to follow the segment specified in the associated #IDENTITY parameter. If it is required that the new segment(s) should be the first in the program, then SEGMENTNAME may be omitted from the #IDENTITY parameter.
12  #DELSEG

This causes one or more segments to be deleted from the file.

The parameter has the following format:

```
#DELSEG N
```

where

\[ N = \text{the number of consecutive segments to be deleted from the file, starting with the one named in the associated \#IDENTITY parameter. N is expressed in decimal, in the range 1 to 511; it may be omitted, in which case one segment will be deleted.} \]

13  #REPSEG

This causes one or more segments to be deleted and replaced by one or more new segments. The number of replacement segments need not necessarily be the same as the number of segments deleted.

The parameter has the following format:

```
#REPSEG N,V
```

where each of \( N \) and \( V \) is optional. \( N \) is as described under \#DELSEG. \( V \) is as described under \#NEWSEG. If both these fields are present they may be in either order, separated by a comma. The comma is only required if both are present.

The effect of \#REPSEG is that of a \#DELSEG followed by a \#NEWSEG.

If \#REPSEG is present, the last of the control parameters for this amendment is followed by the replacement segments and/or parameters, as described under \#NEWPROG.

14  #LINEDIT

This introduces a set of detailed amendments to a source segment.

The parameter has the following format:

```
#LINEDIT
```

with the operand field blank.

If \#LINEDIT is present, the last of the control parameters for this amendment is followed by a set of detailed amendments, on paper tape and/or cards. Each detailed amendment is specified by a \#ALTER line. Source lines to be deleted, or to be preceded by insertions, are referenced by the line number, starting with 1 for the first \#PROGRAM line, as in the compilation lists.

Where lines are to be deleted, the \#ALTER line can have either of the following formats:

```
#ALTER A,N
#ALTER A-B
```

where

\[ A = \text{the line number of the first line to be deleted.} \]

\[ N = \text{the number of lines to be deleted.} \]

\[ B = \text{the line number of the last line to be deleted.} \]

The \#ALTER line will be followed by any lines which are to replace the deleted lines, in normal PLAN source form. All lines following the \#ALTER line will be inserted, until the next \#ALTER, \#FINISH or \#IDENTITY, or the file terminator, is encountered.

Where lines are to be inserted, without any deletions being made, the \#ALTER line has the following format:

```
#ALTER A
```

where

\[ A = \text{the line number of the line in front of which the new lines are to be inserted.} \]

The \#ALTER line will, as before, be followed by the new lines in normal PLAN source form.

Examples:

(a)  \#ALTER 50

This causes the lines following \#ALTER to be inserted, the first new line becoming line 50.
(b)  #ALTER  50,11
This causes the eleven lines starting with line 50 to be deleted, and any new lines following #ALTER to be inserted in their place.

(c)  #ALTER  50-60
This has exactly the same effect as (b).

The line numbers specified in the operand fields of successive #ALTER lines must be in rising sequence within the segment.

15  #COMMENT (#XPMZ only)
This may be used only in copy-and-amend mode. It introduces a set of amendments affecting only the comments on PLAN source lines. If an attempt is made to use a #COMMENT parameter in a run using #XPMX, it and all the associated amendment lines will be flagged and ignored.

The parameter has the following format:

#COMMENT

with the operand field blank.

If #COMMENT is present, the last of the control parameters for this amendment is followed by a set of detailed amendments, on paper tape and/or cards. Each detailed amendment is specified by a #ALTER line. Source lines affected are referenced by line number, starting with 1 for the first #PROGRAM line, as in the compilation lists.

Where comment is to be deleted from source lines, the #ALTER line can have either of the following formats:

#ALTER   A,N
#ALTER   A-B

where

A = the line number of the first line from which comment is to be deleted.
N = the number of lines from which comment is to be deleted.
B = the line number of the last line from which comment is to be deleted.

If the comment removed from these lines is to be replaced by new comment, then the #ALTER line may be followed by lines containing the new comment, in the format described below. The comment from these lines will be incorporated into the successive source lines, starting with line A. If there are more new comment lines than lines from which comment has been deleted, the excess comment lines will overwrite the comment portion of lines B+1 et seq. on the source subfile. However, it is not likely that the above formats will be used in these latter circumstances, for if it is desired to overwrite comment rather than deleting it, the following format may be used:

#ALTER   A

where

A = the line number of the first of the lines whose comment portions are to be overwritten by the comment portions of the lines following the #ALTER line.

A series of new comment lines should follow. The lines containing the new comment must start with a left-hand square bracket in character position 36 of a paper tape line or column 36 of a card. The comment to be deleted or overwritten must start with a left-hand square bracket in position 36 or after; it is starts after position 36, all character positions between position 35 and the bracket must be spaces. If a line to be amended contains comment starting before position 36, or an operand extending beyond position 35, the comment will not be deleted or overwritten. Any new comment line intended for such a source line will be ignored.

The line numbers specified in the operand fields of successive #ALTER lines must be in rising sequence within the segment.

16  #DELAMEND (#XPMZ only)
This causes the amendments subfile of the PLAN source segment specified by the associated #IDENTITY parameter to be deleted. The segment thus returns to the form in which it was written to the file when the file was last reorganized by a copy-and-amend run or, if there has been no copy-and-amend run, when the file was created. If this parameter is input in a #XPMX run, it will be flagged and ignored.
The parameter has the following format:

```
#DELAMEND
```

with the operand field blank

A #DELAMEND parameter may appear under the same #IDENTITY as a #LINEEDIT parameter. The effect of this combination is that the existing amendments subfile for the specified segment is deleted, and the new #LINEEDIT amendments are done to the original version of the segment as it was when the file was either created by #XPMY or last reorganized by an update run in copy-and-amend mode.

As already indicated, if the amendment type is #DELSEG or #DELAMEND, or if the amendment type parameter is omitted, then the amendment is complete without any following amendment data. For any other amendment type, the last of the control parameters is followed by the appropriate amendment data.

A #FINISH parameter should be input at the end of each complete amendment. This parameter makes a pair with #IDENTITY, defining a complete amendment to one program or to one segment. It may be omitted; but it provides additional security if a file is shared between different users.

A #SWITCH directive may be input wherever required to change the input medium from paper tape to cards, or vice versa.

**Batch terminator**

The batch of amendments with their control parameters must be terminated by one of the following parameters:

```
#STOP
```

or

```
#CHAIN
```

#CHAIN introduces a set of parameters for automatic compilation, (see below).

#STOP acts as an input file terminator. It causes the editor program to release its paper tape reader or card reader and to enter its end routines.

**Automatic compilation**

If a #XPMX or #XPMZ update run is to be followed immediately by a compilation run using #XPLX or #XPLZ, it is possible to input parameters to the editor which will cause the compiler to be entered automatically, with no operator intervention.

The parameters for this facility and the rules governing their use are as described for #XPMY (see page 8).

**Programs and Segments from Magnetic Tape (#XPMZ only)**

New or replacement programs or segments may be brought into the updated DISC COSY file from magnetic tape files. The program or segments to be incorporated by an amendment are specified in a #MFILE parameter, and the magnetic tape file from which they are to be copied is specified in an associated IN parameter. If the parameters are input in a #XPMX run, they will be flagged and ignored.

These parameters will form part of an amendment in the paper tape and/or card amendments file, and must follow any other parameters controlling the particular amendment (except that #FINISH, if present, will follow them). They will therefore be preceded by at least a #IDENTITY parameter and either a #NEWPROG, a #REPPROG, a #NEWSEG or a #REPSEG parameter.

Magnetic tape files to be used as input to #XPMZ must conform to the same formats as specified for #XPMY: please refer to page 8 of this chapter.

The format of the #MFILE parameter and its associated parameters is:

```
#MFILE SUBFILENAME/SEGMENTNAME,N
IN(FILENAME(FGN/RSN),TSN)
```

optionally followed by

```
#CONTINUE
```

or
#REWIND

or

#CLOSE

The significances of the various fields in these parameters, and the rules governing their use, are the same as in #XPMY; please refer to page 9 of this chapter.

Programs and Segments from Other Disc Files (#XPMZ only)

New or replacement programs or segments may be brought into the updated DISC COSY file from other disc files. The program or segments to be incorporated by an amendment are specified in a #DFILE parameter, and the disc file from which they are to be copied is specified in an associated INE or INF parameter. If these parameters are input in a #XPMX run, they will be flagged and ignored.

These parameters will form part of an amendment in the paper tape and/or card amendments file, and must follow any other parameters controlling the particular amendment (except that #FINISH, if present, will follow them). They will therefore be preceded by at least a #IDENTITY parameter and either a #NEWPROG, a #REPPROG, a #NEWSEG or a #REPSEG parameter. Disc files to be used as subsidiary input to #XPMZ must be in the format produced by the DISC COSY editor programs.

The format of the #DFILE parameter and its associated parameters is:

#DFILE SUBFILENAME/SEGMENTNAME,N

INE(FILENAME(FGN),CSN) or INF(FILENAME(FGN))

optionally followed by

#CONTINUE

where

SUBFILENAME = the name of the subfile which contains the new segment(s). It may be up to 12 characters long; if it is punched with fewer than 12 characters the program will supply spaces at the right-hand end to complete the 12 characters. Internal spaces will not be removed and will be included in the count of 12 characters.

SEGMENTNAME = the name of the first segment to be copied from the specified subfile under the control of this parameter. It may be up to 11 characters long; if it is punched with fewer than 11 characters the program will supply spaces at the right-hand end to complete 11 characters. In the case of the segment name, the program will remove any internal spaces and disregard them in the count of 11 characters (this is to conform with compiler practice). SEGMENTNAME may be omitted, in which case the preceding solidus and the N must also be omitted. If it is omitted, the whole of the subfile will be copied.

N = the number of segments to be copied, starting with the one specified by SEGMENTNAME; in decimal, in the range 1 to 511. It may be omitted, in which case the preceding comma should also be omitted. If it is omitted (and SEGMENTNAME is present) only one segment will be copied.

FILENAME FGN CSN

Together specify the disc file containing the subfile specified by the #DFILE parameter. Their significances and the rules governing their use are as described for INE or INF under 'Initial Control Parameters'.

#CONTINUE should be used if further segments are to be copied from the same disc file in a later amendment. If #CONTINUE is used, the file will be left open, and an INE or INF parameter will not be required following the next #DFILE. If #CONTINUE is not used the file will be closed after the required segments have been copied.

When inserting a new PLAN source segment from an additional input file on disc, it is possible to do detailed amendments to it at the same time. To do this the #DFILE set of parameters should be followed immediately by either a #LINEDIT or a #COMMENT parameter introducing the set of detailed ALTER amendments. These detailed amendments must be terminated by a parameter consisting of a single asterisk punched in the first character position.

If a #DFILE parameter specifies either a number of consecutive segments or a complete subfile, detailed amendments may still follow, and will be taken as referring to the first of the segments specified.

Detailed amendments may follow a #DFILE set of parameters even if the latter appears as only part of the amendment data for a #NEWPROG, #REPPROG, #NEWSEG or #REPSEG amendment.
The following is an example of the use of this facility. These parameters would have the effect of creating a new 3-segment program with the fourth segment being input from a basic peripheral, the other segments being input from an additional input file on disc, and the second segment being amended.

```
IDENTITY    NEWSUBFILE
NEWPROG
STEER       LIST, OBJECT
DFILE       OLDSUBF/SEG1
INSERT FILE (2)
CONTINUE
DFILE       OLDSUBF/SEG2,2
CONTINUE
LINEDIT
ALTER       13,2
line amendments
*
PROGRAM     /SEG4
source lines
END
DFILE       OLDSUBF/SEG5
FINISH
```

Line Printer Output

All parameters are listed on the line printer as they are read; so, too, are the lines of amendment data, except that in the case of #NEWPROG, #REPPROG, #NEWSEG and #REPSEG only the first line of each inserted segment is printed. Any parameter lines which are in error are flagged with two asterisks in the left-hand margin. Any parameters or data lines which contain a paper tape format error are printed flagged with a *P* in the left-hand margin. Certain error messages may also be output on the line printer while the file is being updated.

If #NEWPROG, #REPPROG, #NEWSEG or #REPSEG is present with a V operand, a list of any source lines that are out of sequence will also be printed and flagged with *P*, as explained earlier.

As the amendments to each program are completed, the editor program prints a summary listing of the segments in that program's subfile. Apart from the heading "Updated by #XPMX" or "Updated by #XPMZ", this listing is similar in format to that produced by #XPMY, described on pages 10 and 11 of this chapter.

The third column of the summary segment gives an indication of the degree of complexity reached in the organization of the amendments subfile. It shows the number of times it has been necessary to extend each segment's amendments subfile. As each extension will not be in buckets adjacent to the original subfile, the subfile is then said to be comprised of a number of fragments, and it is the number of fragments forming the associated amendments subfile which is printed in the third column of the segment summary. The DISC COSY compilers will not accept more than 10 fragments per amendments subfile, so when any of the figures in this column approaches 10 it is an indication that a copy-and-amend run should be undertaken to reorganize the file. However, for the program to operate at maximum efficiency a reorganization should be undertaken before this stage is reached.

After the summary listing of the last subfile amended, the editor program prints a list of the subfiles containing unamended programs, followed by a final summary of the file, again in a format similar to that produced by #XPMY, described on page 11 of this chapter.
Line Printer Error Messages

The following error messages may be output on the line printer:

**APPEND FILE CANNOT BE RENAMED**

A REN parameter has been input with an APP parameter. A REN parameter should only be used with an OUT parameter.

**SUBFILE name NOT FOUND**

The subfile specified in a #IDENTITY parameter cannot be found in the input file being updated; or the subfile specified in a #MFILE or a #DFILE parameter cannot be found in the subsidiary input file specified by the associated IN or INE or INF parameter. The program proceeds to the next amendment.

**SUBFILE name ALREADY AMENDED**

The subfile specified in a #IDENTITY parameter was specified in a previous #IDENTITY parameter in this run, but the amendment immediately preceding this amendment related to a different subfile. This amendment is rejected and the program proceeds to the next amendment.

**DUPLICATE subfilename REJECTED**

The amendment type is #NEWPROG and the subfile name specified by the #IDENTITY parameter is already in use in the file. The amendment is rejected and the program proceeds to the next amendment.

**NO MORE PROGRAMS CAN BE ACCEPTED**

During a copy-and-amend run, 33 programs have already been copied to the output file and the message is printed out during the listing of unamended programs before the first rejected program, or, a further program has been introduced to the file by a #NEWPROG and the message is output during an amendment to a subsequent program which now happens to be the 34th. program in the file. During either a copy-and-amend or an amend-in-situ run, the amendment type is #NEWPROG but there are already 33 programs in the file; this amendment is rejected and the program proceeds with the next amendment.

**SEGMENT name NOT ON MASTER FILE**

The segment named in a #IDENTITY parameter cannot be found on the file. The amendment is rejected.

**SEMI-COMPILED CANNOT BE EDITED**

The segment specified in a #IDENTITY parameter is in semi-compiled form, and the amendment type is #INEDIT, #COMMENT or #DEAMEND. The amendment is rejected.

**TOO MANY SEGMENTS BEING DELETED**

The number in the operand of a #DELS or #REPSEG parameter is greater than the number of segments from the one specified to the end of the subfile. All the segments from the one specified to the end of the subfile are deleted.

**AMENDMENTS ATTEMPTED AFTER #END**

This message can only occur in a copy-and-amend run. The program is merging an amendments subfile with the corresponding source subfile and finds that a tag is pointing to a position after the #END. The amendments up to this point are done correctly, but any deletions or insertions after #END are ignored.

**PREVIOUS RUN ABANDONED**

A copy-and-amend or amend-in-situ run has been attempted but the file has been corrupted by abandoning a previous amend-in-situ run. The corrupted file should be recreated from the previous dump and brought up-to-date either by using tag dumps or by carrying out any outstanding amendments.

**SEGMENT name NOT FOUND**

These messages may occur when the user is trying to copy segments from another disc or magnetic tape file. The program abandons the attempt and proceeds to the next amendment.

A number of segments has been specified in a #MFILE or a #DFILE parameter, but the end of the subfile has been reached before the specified number of segments has been copied. The number n shows the shortfall in the number of segments. The program proceeds to the next amendment.
OUTPUT TAPE FULL LAST SEGMENT name

NO #END SUPPLIED, SEGMENT GIVEN ONE.

The end of tape marker has been detected on the magnetic tape specified by the OUT or APP parameter. The tape is backspaced to the end of the last complete segment, and the end-of-subfile sentinels and end-of-composite-file trailer label are written.

The amendment type is #NEWPROG, #REPPOG, #NEWSEG or #REPSEG, and a #FINISH, a #STOP or a #IDENTITY has been read, but no #END for the current segment has been detected. The program infers a #END for the current segment, and continues.

Operating Instructions for #XPMX or #XPMZ

1 If exchangeable disc files are used, load the cartridge or cartridges containing:
   (a) the required editor program, #XPMX or #XPMZ.
   (b) the required compiler, if the compiler parameter is present.
   (c) the file to be updated.
   (d) the work file.
   (e) the output file (copy-and-amend mode).
   (f) any subsidiary exchangeable disc input files (#XPMZ only).

2 Load any magnetic tapes that are required for input or output (#XPMZ only).

3 Input the message:
   
   FI#XPM- #NAME
   
   where NAME represents the last four characters of the name of the file containing #XPM-.

4 Load the parameters and data in the paper tape reader and/or card reader.

5 If it is required that the editor program should not output a display message to the console typewriter for each subfile it amends, input the console message:

   ON #XPM- 20

6 If the initial control parameters are on paper tape, activate by:

   GO #XPM- 20

   If the initial control parameters are on cards, activate by:

   GO #XPM- 21

7 The program allocates the appropriate reader and the line printer, opens and closes or unloads disc files and magnetic tape files as required, and updates the file.

8 After completing all the amendments for each subfile, the editor program outputs one of the following console messages:
   (a) If all the amendments were completed successfully, and no parameter errors were detected:

   0#XPM-; DISPLAY=- SERV OK SUBFILENAME

   (b) If any parameter errors were detected or if any line printer error messages were output for any reason:

   0#XPM-; DISPLAY=- SERV ER SUBFILENAME

   These console messages may, if required, be inhibited by setting a switch before entering the editor program (see operating instruction number 5, above).

9 When the run is completed the program deletes itself with a console message in the form:
   (a) If no compiler parameter was present:

      0#XPMX; DELETED=- MX

   or

      0#XPMZ; DELETED=- MZ

4322(7.72)  Chapter 11  33
(b) If a compiler parameter was present:

\[ \text{0\#XPM-;} \text{, DELATED: FI \#ABCD \#NAME N} \]

where \#ABCD, \#NAME N are as specified in the compiler parameter.

**EXCEPTION CONDITIONS**

The following messages may be output on the console typewriter during the running of the program:

1. **0\#XPM-; HALTED:-- INCORRECT FILE PARAMETERS. FIX & RESTART**
   
   This message is output if there is a format error in any of the initial control parameters, or if exchangeable disc and fixed disc versions of these parameters are used together. The parameters should be corrected and the program restarted from step 4.

2. **0\#XPM-; HALTED:-- NEEDS TAPE NNNN or 0\#XPM-; HALTED:-- NEEDS DISC NNNN**
   
   These messages are output if a tape serial number or a cartridge serial number respectively has been checked and found to be incorrect. The correct tape or cartridge as specified in the appropriate parameter should be loaded, and the program resumed by:

   \[ \text{GO \#XPM-} \]

3. **0\#XPM-; HALTED:-- FILE name NOT FOUND**
   
   This message is output if the file specified by the INE or INF parameter or the OUTE or OUTF parameter cannot be found. Either the correct cartridge should be loaded and the program resumed by:

   \[ \text{GO \#XPM-} \]
   
or the parameter should be corrected and the program restarted by:

   \[ \text{GO \#XPM- 20 or GO \#XPM- 21} \]

4. **0\#XPM-; HALTED:-- WORK FILE NOT FOUND**
   
   This message is output if:

   (a) the permanent file specified by the OUTW parameter cannot be found; or

   (b) a scratch file cannot be opened on the cartridge or unit specified by the OUTW parameter, where this specifies a scratch file; or

   (c) a scratch file cannot be opened on the cartridge or unit containing the output file, if no OUTW parameter is present.

   If an OUTW parameter is present, either the correct cartridge should be loaded and the program resumed by:

   \[ \text{GO \#XPM-} \]
   
or the parameter should be corrected and the program restarted by:

   \[ \text{GO \#XPM- 20 or GO \#XPM- 21} \]

   as appropriate.

   Alternatively, or if no OUTW parameter is present, the console message:

   \[ \text{GO \#XPM- 29} \]

   will enable the restrictions imposed by the presence or absence of an OUTW parameter to be disregarded, and Executive to search for a scratch file on any cartridge or unit which is on line. If the same output message then occurs, it may be possible to make more space available by deleting another program which is using disc scratch files, or, if exchangeable discs are in use and more than one transport is available, by loading another cartridge. The message:

   \[ \text{GO \#XPM- 29} \]

   can then be repeated. If still unsuccessful the run must be abandoned.
One of these messages may occur when the program is seeking to extend the indicated file. If it is possible to make more space available on the cartridge or unit by deleting another program which is using disc scratch files, then after doing so the message:

```
GO #XPM-
```

will cause the program to make another attempt to extend the file.
Alternatively, if more than one transport or unit is available, the message:

```
AL #XPM- 7 *NNNNNN
```

(where NNNNNN represents the octal serial number of an alternative cartridge or unit) followed by:

```
GO #XPM- 27
```

will cause the file to be extended onto the second cartridge or unit, if space is available. If space on the second cartridge or unit is not available, the appropriate HALTED message will again occur.

6 0#XPMZ; HALTED:-- ADDITIONAL INPUT FILE NOT FOUND

This message is output if the file specified by an INE or INF parameter associated with a #DFILE parameter cannot be found. The correct cartridge should be loaded, or the parameter corrected, as the case may be, and the program resumed by:

```
GO #XPMZ
```

If the cartridge cannot be made available the console message:

```
GO #XPMZ 28
```

will cause the program to continue the run, omitting any segments that were to have been copied from the file.

7 0#XPM--; DISPLAY:-- NEEDS DISC NNNN

This message is output immediately before message 3, 4 or 6 above, if the parameter specifying the file to be opened included a CSN field.

8 0#XPMZ; LOAD FILENAME

where FILENAME is the name of a magnetic tape input file specified by an IN parameter associated with a #MFILE parameter. Make the tape available and resume by:

```
GO #XPMZ
```

If the tape cannot be made available, the console message:

```
GO #XPMZ 28
```

will cause the program to continue the run, omitting any segments that were to have been copied from the file.

9 0#XPMZ; DISPLAY:-- OUTPUT TAPE FULL

This message is output if the end-of-tape marker is sensed on the magnetic tape specified by an OUT or APP parameter. A fuller message is output to the line printer, giving the name of the last segment written to the tape.

10 0#XPMZ; DISPLAY:-- TAPE MISREAD

This message is output if a misread is detected while the program is copying a segment onto the output file from a magnetic tape file. The message is accompanied by one of the following line printer messages:

```
MT BLOCK MISREAD. SEGMENT REJECTED.
```

The segment in which the misread block occurs is rejected.

```
INVALID RECORD LENGTH. SEGMENT REJECTED.
```

The segment in which the invalid record length occurs is rejected.

```
INCORRECT BLOCK COUNT.
```
The new segment(s) are not rejected.

INCORRECT CHECKSUM.

The new segment(s) are not rejected.

The run continues, omitting any rejected segments.

11 0#XPM-; HALTED:-- INPUT FILE IS IN WRONG FORMAT

This message is output if the file specified by the INE or INF parameter is not in the correct subfile format. Abandon the run.

12 0#XPM-; DISPLAY:-- PL

This message is output if a paper low condition has been detected on the line printer. The line printer is disengaged. The paper should be replenished and the line printer reallocated to continue the run.

SECURITY DUMPS OF DISC COSY FILES

On a machine with 16K or more words of core store, #XPMZ may be used for updating, and security dumps may be obtained during the update run by using #COPYPROG, #COPYSEG, #DUMPPROG and #DUMPSEG parameters (see pages 24 and 25).

On a machine with 8K words of core store, #XPMX must be used for updating, and this does not include facilities for providing security dumps. The utility program #XMQ is therefore provided for use in conjunction with #XPMX, for obtaining dumps of specified segments and amendments subfiles from DISC COSY files. The program as supplied has a priority of 80.

Hardware requirements for #XMQ

4K words of core store

1 line printer

1 paper tape reader or 1 card reader

1 E.D.S. or 1 F.D.S.

1 card punch (optional)

1 paper tape punch (optional)

1 magnetic tape deck (optional)

Basic Peripheral usage, #XMQ

The program reads its parameters from either paper tape or cards depending on the entry point used. These parameters define the E.D.S. or F.D.S. file to be opened as input, which segments or amendments subfiles are to be dumped, and onto which peripherals.

The paper tape or card reader and the line printer are allocated at the beginning of the run and released at the end. The card punch and paper tape punch are allocated if and when first required and retained until the end of the run.

Magnetic Device Usage, #XMQ

The medium used, exchangeable disc or fixed disc, is specified by the parameters, as described under 'Control Parameters' below.

OBLIGATORY DISC FILE

The input file is opened as ED0 or FD0 after the initial control parameters have been read and is closed at the end of the run.

OPTIONAL MAGNETIC TAPE

If segments are to be dumped to magnetic tape, the tape is opened as MT0 when it is required and retained until the end of the run.
Input

The input file on disc must be in the format created by one of the DISC COSY programs \#XPMX, \#XPMY or \#XPMZ.

Input parameters must all be punched starting in column 1 of a card or in the first character position of a paper tape record.

Output

PLAN source and semi-compiled segments can be output on cards, paper tape, magnetic tape or on the line printer. Amendments subfiles can be dumped onto cards or paper tape only.

When dumped to cards, PLAN source and semi-compiled segments will have sequencing inserted. This sequencing will be in tens, and the identity field will consist of the first 3 characters of the segment name. A blank card will be punched at the beginning and end of each segment. In any amendments subfile dump on cards, the card sequence numbers on some of the records will be one less than they should be. However the dump will be acceptable to \#XPMY.

Dumps of segments on paper tape will not have sequencing inserted. Existing card sequencing will be removed from PLAN source segments, as will any complete words of trailing spaces in the operand. Horizontal tabs will not be inserted. Three run out characters will be punched between records and six inches of run out characters at the beginning and end of each segment.

However, if a DUMP parameter is input specifying output to both cards and paper tape, the amendments subfile dump on cards will be incorrect and will not be acceptable as input to \#XPMY. If it is required to dump the same amendments subfile to both peripherals, two separate DUMP parameters should be input.

Each segment dumped on the line printer will start on a new page and lines will be numbered down the left-hand side. If a segment is being dumped on cards at the same time, the line printer listing will include the new card sequencing; otherwise, it will include any card sequence fields existing on the disc file.

When output to magnetic tape, PLAN source segments are put in one composite subfile on magnetic tape and this subfile is given the same name as the composite subfile containing the program on the disc file. Consecutive segments of either PLAN source or semi-compiled are put in the same level 1 subfile. Magnetic tape files produced by \#XPMQ are in the correct format for input to \#XPMX, \#XPMY and \#XPMZ. They are also acceptable to \#XPMR and \#XPMS provided that any semi-compiled segments come after all the PLAN source segments in a program.

Amendments subfiles are dumped with one card or record of paper tape containing either a tag or an inserted source line. Sequencing is inserted on cards. The first and last record for each amendments subfile are a \#TAG and a \#FINISH with formats as described on page 24 of this chapter.

Control parameters

These consist of one or more initial control parameters followed by any number of parameters defining which segments are to be dumped and on which peripherals. These are followed by a terminating parameter.

The initial control parameters define the disc file to be opened as input, and optionally a magnetic tape to be opened as output and the program to be brought into store when the run is completed. They may appear in any order before the first of the next set of parameters, and have the following format:

1 \{\text{INE} (\text{FILENAME} (\text{FGN}), \text{CSN}) \}

or

\text{INF} (\text{FILENAME} (\text{FGN}))

where the significance of the various fields and the rules governing their use are as described on page 17 of this chapter.

This parameter is compulsory and defines the COSY file on E.D.S. or F.D.S. containing the segments to be dumped.

2 \{\text{OUT} (\text{FILENAME} (\text{FGN/RSN}), \text{TSN}) \}

or

\text{APP} (\text{FILENAME} (\text{FGN/RSN}), \text{TSN})


This parameter is necessary only if some segments are to be dumped to magnetic tape. It defines the tape to be opened. If the first form is used, the dump will overwrite any information already on the tape.

If the second form is used, the tape must already be in subfile format, and any subfiles dumped will be appended onto the existing contents of the tape.

3 REN(FILENAME(FGN/RSN,RET))

This parameter is optional. It should be input only if an OUT parameter is input, and it causes the specified tape to be renamed.

4 NEXT (#XXXX #YYYY nnnn)

This parameter is optional. If it is present, #XPMQ will delete itself and find the specified program when the run has finished. Thus if the above parameter was present, the run would end with the message:

DELETED:-- FI #XXXX #YYYY nnnn

The contents of the brackets may include any fields acceptable by Executive after the 'FI' in the FIND message, but must not exceed 37 characters. If this parameter is not present, the run will end with the message:

DELETED:-- HH

The parameters defining the segments to be dumped and the peripherals to be used have the following format:

TYPE (XX,YY,ZZ) SUBFILENAME/SEGMENTNAME(M),N

where

TYPE is either COPY or DUMP, COPY meaning output complete segments and DUMP, output amendment subfiles only.

XX,YY,ZZ are the symbolic names of the peripherals on which the segments or amendment subfiles are to be dumped. These may be any combination of CP, TP, LP and MT in any order separated by commas, except that LP and MT may only be used in a COPY parameter, and will be ignored in a DUMP parameter.

SUBFILENAME must be present and defines the name of the composite subfile containing the segments or amendment subfiles to be dumped. Internal spaces are significant and the field has a maximum length of 12 characters. If fewer than this are punched the program will supply trailing spaces.

SEGMENTNAME defines the name of the segment, or of the first of a number of consecutive segments, for which a dump is required. It may be omitted together with the preceding solidus and the M and N fields, in which case a dump will be given of all the segments or amendment subfiles in the specified composite subfile.

M is a number specifying to which version of the specified segment the parameter refers. It should be used if the composite subfile specified contains more than one segment with a given name, and indicates that the program is to look for the Mth version. M may be omitted together with the brackets round it, where the first or only version of the named segment is being specified.

N specifies the number of consecutive segments, starting with the one specified by SEGMENTNAME(M), for which a dump is required. It may be omitted, together with the preceding comma, in which case the dump will be given for one segment only.

The only restrictions on the format of this parameter are that the TYPE field must start in column 1, and that the SUBFILENAME field must be separated from the bracket at the end of the peripherals field by at least one space.

Any number of these parameters may be input to #XPMQ, and subfiles and segments on the input file may be referenced in any order.

If the parameter type is DUMP, the program will look at all the specified segments but will produce output for only those with amendments subfiles.

If a segment specified in a COPY parameter is a PLAN source segment with an associated amendments subfile, it will be dumped out with the amendments incorporated.

The terminating parameter has the following format:

PEND

This parameter should follow the last of the parameters specifying segments to be dumped, and will therefore always be the last parameter input.
Example of control parameters
INE(COSYSRCFILE(3), 717032)
OUT(DUMPTAPE, 100242)
NEXT(#XPLZ #SEEK)
COPY(MT) FRED
COPY(LP, CP) FRED/SEG4
DUMP(CP) DICK/SEG2,4
DUMP(CP) DICK/SEG2(2),4
PEND
This set of parameters would cause #XMQ to
1 Open the specified E.D.S. file as input
2 Dump all the segments in subfile FRED onto the specified magnetic tape, and SEG4 onto cards and the
   line printer as well
3 Look at the four segments in subfile DICK starting with the first version of SEG2, and the four starting with
   the second version of SEG2, and then dump any amendments subfiles associated with these segments onto
   cards
4 End with the message
   DELETED:-- FI #XPLZ #SEEK

Line Printer Output
The first line printed is always a heading giving the date of the run and the mark number of #XMQ being used.
The format of this heading is as follows:
   PRODUCED BY #XMQ 1 07/05/69
All input parameters are listed as they are read. Any unrecognized parameters or parameters in the wrong format
are flagged with two asterisks in the left-hand margin.
The format of segments dumped to the line printer is described under Description.

Line Printer Error Messages
The following error messages may be output to the line printer during a #XMQ run:
1 APPEND FILE CANNOT BE RENAMED
   A REN parameter has been input but the magnetic tape has been specified by an APP parameter. A REN
   should only be used if the tape is specified by an OUT parameter.
2 SUBFILE name NOT FOUND
3 SEGMENT name NOT FOUND
   The name specified in a COPY or DUMP parameter cannot be found on the input file. The run continues
   normally with the next parameter.
4 n SEGMENTS NOT FOUND
   A number of segments has been specified in a COPY or DUMP parameter, but the end of the subfile has been
   reached before the end of the specified segments. The number n specifies the shortfall.
5 COPY IGNORED AS NO MT OPENED
   A COPY parameter has been input specifying output to magnetic tape, but no OUT or APP parameter was
   input. If the COPY parameter specifies other peripheral types in addition to MT, it will be obeyed. Otherwise,
   the program will continue with the next parameter.
6 segmentname HAS NO AMENDMENT SUBFILE
   A segment specified in a DUMP parameter has not got an amendments subfile associated with it. No output
   is produced for this segment but the run continues normally.
AMENDMENTS ATTEMPTED AFTER END

A segment specified in a COPY parameter has an amendments subfile containing a tag pointing beyond the END record of the segment. A END is output and the run continues normally.

TAPE FULL; name LAST SEGMENT OUTPUT

The end of tape marker has been detected on the magnetic tape used for output. The program backspaces to the end of the last complete segment and writes the end of subfile sentinels and the trailer label. The run continues, ignoring any further requests to output to magnetic tape.

segmentname IS WRONG SUBFILE TYPE

The subfile containing the specified segment has been found, but the type of subfile indicates that it contains data, other than PLAN source or semi-compiled. The run continues with the next segment specified.

FILE NOT IN SUBFILE FORMAT

The specified file on E.D.S. or F.D.S. is not a composite file. No output is produced and the run stops with the console message:

0#XPMQ; HALTED:- ZZ

segmentname IS IN WRONG FORMAT

The specified segment is not in DISC COSY format.

UNACCEPTABLE RECORD

A record has been found whose length is not in the range 1 to 21 words. The run stops with the console message:

0#XPMQ; HALTED:- ZZ

Operating Instructions

1 If E.D.S. files are being used, load the cartridge or cartridges containing
   (a) the program #XPMQ
   (b) the COSY file to be used as input

2 Input the message
   FL #XPMQ #NAME

   where the name of the file containing #XPMQ is PROGRAM NAME.

3 If necessary, load the magnetic tape to be used as output.

4 Load the parameters in the paper tape or card reader.

5 If the parameters are on paper tape, activate by
   GO #XPMQ 20

   If the parameters are on cards, activate by
   GO #XPMQ 21

6 When the run is complete, the program deletes itself with a message of the form
   (a) if no NEXT parameter was input
      0#XPMQ; DELETED:- HH
   (b) if a NEXT parameter was present
      0#XPMQ; DELETED:- FL #XXXX #YYYY nnnn

   where all the fields after the 'FL' are as specified in the NEXT parameter.
EXCEPTION CONDITIONS

The following error messages may be output to the console typewriter during the course of a run:

1  0#XPMQ; HALTED: - INCORRECT FILE PARAMETERS, FIX & RESTART.
   Either no INE or INF is present, or one of the initial control parameters is unacceptable. The parameters
   should be corrected and the program restarted from step 4.

2  (a) 0#XPMQ; DISPLAY: - NEEDS DISC *NNNNN
    (b) 0#XPMQ; HALTED: - FILE name NOT FOUND
   Message (b) indicates that the file specified by the INE or INF parameter cannot be opened. It is accompanied
   by message (a) only if the parameter specified the cartridge serial number. Either the correct cartridge should
   be loaded and the program resumed by:
   
   GO #XPMQ
   
   or the parameter should be corrected and the program restarted from step 4.

3  0#XPMQ; HALTED: - NEEDS DISC *NNNNNN

0#XPMQ; HALTED: - NEEDS TAPE *NNNNNN

   These messages are output if a cartridge serial number or tape serial number respectively has been checked
   and found to be incorrect. The correct cartridge or tape should be loaded and the program resumed by:
   
   GO #XPMQ

4  0#XPMQ; DISPLAY: - TAPE FULL
   The end of tape marker has been detected on the magnetic tape used for output. A fuller message is output
   to the line printer, giving the name of the last segment dumped.

5  0#XPMQ; HALTED: - ZZ
   The run is unable to continue for some reason; for example, the input file is in the wrong format. The reason
   for the halt is printed on the line printer. The run should either be abandoned or restarted using a different
   set of parameters.

RECREATING A DISC COSY FILE

It is recommended that, as a file security measure, a dump of the complete file should be obtained on paper tape,
cards or magnetic tape, every time a DISC COSY file is reorganized by a copy-and-amend run. On a processor with
16K or more words of core store, this dump can be obtained during a #XPMZ update run by the use of a
#COPYPROG parameter for each program being amended during the run, together with a dummy amendment
(a #IDENTITY, a #COPYPROG and a #FINISH parameter) for each program not being amended during the run.
On a processor with 8K words of core store, this dump can be obtained by doing a run using the utility program
#XPMQ.

Similarly, it is recommended that a dump of the amendments subfiles of all programs being amended should be
obtained on paper tape or cards during each amend-in-situ run. This dump can be obtained using #XPMZ on a
processor with 16K or more words of core store, or using #XPMQ on a processor with 8K words of core store.

If these recommendations are adopted, then, should any catastrophe occur to a DISC COSY file, it will always be
possible to recreate the file in its latest amended state by using these dumps as input to #XPMY. If during a
recreate run #XPMY receives as input, for each program in turn, just the dump from the last copy-and-amend
update run followed by the dump from the last relevant amend-in-situ update run, then each program's subfile will
be recreated as last amended, subject to the following qualifications:

1  If, since the last reorganization of the file, any segments have been deleted, they will reappear in the recreated
   subfile.

2  If any new segments have been added since the last reorganization of the file, they will not be present in the
   recreated subfile.

3  If any segments have been replaced since the last reorganization of the file, it is the replaced segments, not the
   replacement segments, that will be present in the recreated subfile.
To include new or replacement segments in the recreated subfile it is necessary to insert them (or, if they are to be obtained from magnetic tape or disc files, their appropriate parameters) on paper tape or cards in the input file for #XPMY. They should precede the amendments subfiles dump in the input file. If the sequence of the segments in the file is important, and the copy-and-amend dump is on paper tape or cards, then the new or replacement segments can be inserted in the appropriate positions in that dump. If the copy-and-amend dump is on magnetic tape, the new or replacement segments can be inserted in the required positions in the recreated subfile by defining the copy-and-amend dump input by a series of #MFILE parameters, except the last with #CONTINUE: the new or replacement segments being interspersed among the parameters. As the subfile of a copy-and-amend dump is given the same subfile name as the composite subfile being dumped, it is this name which should appear as SUBFILENAME in the #MFILE parameters.

To eliminate deleted or replaced segments from the recreated file a similar approach may be adopted. If the copy-and-amend dump is on paper tape or cards, remove the redundant segments from the dump before using it as input to #XPMY. If the copy-and-amend dump is on magnetic tape, declare it in a series of #MFILE parameters, with #CONTINUE (except for the last), omitting to specify the redundant segments.

It is important that the amendments subfiles dump should come at the end of the input for the subfile being recreated, because once a #TAG record has been read no more segments will be accepted. The program tests to see if there is a related segment subfile already written to the recreated file before it writes an amendments subfile, and rejects the latter if there is not.

There is one restriction on the possibility of file recreation by these means. The program will only accept one amendments subfile dump for each segment name; if more than one such dump for any particular segment name is present in the input, only the one with the latest date in the #TAG record is accepted. This dump will be applied to the first source segment of that name. It follows that if there is more than one source segment of a particular name, then there is a danger that related amendments subfiles may be omitted from the recreated subfile, and that an amendments subfile may be applied to the wrong segment.

If any programs have been deleted from the file since the last copy-and-amend run, neither their #COPYPROG dumps nor any #DUMPROG dumps relating to them should be included in the input for the file recreate run.

If any programs have been replaced since the last copy-and-amend run, then their subfiles #COPYPROG dumps should not be included in the input for the file recreate run. Instead, the replacement programs should be brought in from their original medium, that is, in the form in which they originally accompanied their #REPPROG parameters. For each such program, a #DUMPROG dump should be included in the input only if it was obtained subsequently to the relevant #REPPROG run.

**Example of an Input File for File Recreation**

This example assumes that it is necessary to recreate a DISC COSY file on exchangeable disc. For the sake of conciseness it illustrates the recreation of the first program’s subfile only; input for subsequent programs in the file would follow, each program’s input commencing with a #IDENTITY parameter and concluding with a #FINISH parameter in the usual way. The file would terminate with a #STOP parameter.

The first program on the file contained twelve segments. Of these, the third and fourth had been input from cards to replace a deleted segment, and the eighth segment had been inserted from a magnetic tape file; all since the file had last been reorganized.

The last copy-and-amend dump was on magnetic tape and the last amend-in-situ dump was on cards. The #XPMY input file to recreate this DISC COSY file might be as follows:
<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUTE(GASHFILE,770032)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RENE(COSTFILE(2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUTH(WORKAREA,770005)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XPI.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#IDENTITY</td>
<td>PROJECTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#VARIABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#STATE</td>
<td>SHORTLIST,OBJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FILE</td>
<td>PROJECTA/SEGONE,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN(COSFILEDUMP(2),10375)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CONTINUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PROGRAM</td>
<td>PROA70/SEG3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First two segments from copy-and-amend dump tape.

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PROGRAM</td>
<td>PROA70/SEG3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two segments from cards; segment three will not be read from the copy-and-amend dump tape.

Source lines

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FILE</td>
<td>PROJECTA/SEGFOUR,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FILE</td>
<td>OUTPUTS/TABULATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN(FORMATFILE(5),10312)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FILE</td>
<td>PROJECTA/POGTTAB1,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN(COSFILEDUMP(2),10375)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#TAG</td>
<td>/SEGONEWYYYYY,622R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Three more segments from the copy-and-amend dump tape.

Inserted segment from a magnetic tape file.

Etc.

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINISH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Four final segments from the copy-and amend dump tape.

Amend-in-situ dump of amendments subfiles on cards.

Parameters and data to recreate other subfiles.

<table>
<thead>
<tr>
<th>LABEL</th>
<th>OPERATION</th>
<th>ACC</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#STOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 11
Line Printer Output

The line printer output during a file recreation run will be the same as for any other run of #XPMY, as described on page 10, with certain additional features arising from the presence of the amendments subfiles dumps in the input. For each program, before the segment summary, a list of all the #TAG lines appearing in the amendments subfiles dump is printed. If appropriate, one of the following messages may appear on the line following a #TAG line:

LATER VERSION ALREADY INPUT
REPLACES PREVIOUS VERSION
REJECTED. INCORRECT FORMAT
(The operand of #TAG is incorrect)
REJECTED. CARD OUT OF SEQUENCE
(If any card is out of sequence the whole of that segment’s amendments subfile is rejected.)
REJECTED. INCORRECT NO. OF RECORDS
REJECTED. IMPOSSIBLE AMENDMENT SUBFILE
(There is some other error present in the amendments subfile, arising from faulty handling of the cards or paper tape; for example, missing records or damaged records incorrectly repunched.)

THE DISC COSY COMPILERS

The compilers which form part of the DISC COSY compilation system are the PLAN 3 compilers #XPLX and #XPLZ and the PLAN 4 compiler #XPLT. The specifications and operating instructions for #XPLX and #XPLZ are to be found in Chapter 7; those for #XPLT are in Chapter 8.

ADDITIONAL INFORMATION

The foregoing sections of this chapter contain all that the general user requires to know about the DISC COSY compilation system. This section contains additional information on some of the more technical aspects; but it must be stressed that a knowledge of these aspects is not necessary in order to be able to use the system. The right is reserved to make alterations of detail to the information contained in this section.

Subfile Formats

The files produced by the DISC COSY editor programs are composite files each containing up to 33 programs. The structure of one of these files is illustrated in the accompanying diagram. The file contains a directory subfile at level 0, consisting of the directory description and up to 34 subfile descriptions. One of these gives details of any spare space left unused at the end of the composite file; such spare space is regarded as comprising a data subfile. The other subfile descriptions each describe a composite subfile containing one program; that is, they each define a lower level, level 1, directory subfile.

Each level 1 directory subfile contains its directory description and a subfile description for each of its level 2 data subfiles. Each level 2 data subfile contains one segment in either PLAN source or semi-compiled form; there may be as many of these data subfiles as are necessary.

The first record of each source subfile contains steering information for the segment held in that subfile, in the form of a #STEER parameter card image: the operand field information is taken from the #STEER or #SEGSTEER parameter applicable to the segment. Any segment may be held in either or both of PLAN source and semi-compiled forms.

Subfiles may comprise one or more subfile fragments, a fragment being a number of sequential buckets.

From each program, the sequence in which segments will be presented to the compiler, and hence their effective sequence in the file, is determined by the sequence in which their subfile descriptions appear in the level 1 subfile directory. This in turn is determined by the sequence in which the segments (or their controlling parameters, if the segments are being copied from magnetic tape files) appear in the paper tape/card input file to #XPMY. For further segments inserted by an updating run, the sequence is determined by the operand of the #IDENTITY parameter.
Data subfiles are simple subfiles. Directory subfiles are composite subfiles.

It is the level 1 directory subfile whose name is specified by the operand of the \#XPMY \#IDENTITY parameter.

The consolidated program files output by the DISC COSY system have a similar structure, except that they are single-program files, as shown in the diagrams below.

**Disc Consolidator Output File (overlay program)**

The subfile descriptions in the level 1 directory subfile are not necessarily in the order shown.
Disc Consolidator Output File (non-overlay program)

<table>
<thead>
<tr>
<th>DIRECTORY SUBFILE LEVEL 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SUBFILE LEVEL 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational Buckets</td>
</tr>
<tr>
<td>(request and entry blocks)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SUBFILE LEVEL 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprising any</td>
</tr>
<tr>
<td>spare space unused</td>
</tr>
<tr>
<td>at the end of the</td>
</tr>
<tr>
<td>Composite File</td>
</tr>
</tbody>
</table>

No attempt has been made to illustrate fragmented files in the above diagrams.

If during a #XPMY create run a subfile is set up containing no PLAN source or semi-compiled segments, the resultant output file will not be in subfile format. The effect of using such a file as input to #XPMX or #XPMZ is indeterminate.

**Magnetic Tape Input Files**

It has been stated earlier in this chapter that magnetic tapes output by the PLAN compiler and consolidators, and by the magnetic tape COSY editors, are suitable for input to the DISC COSY editors, as are magnetic tape dumps produced by #XPMQ and #XPMZ. Other magnetic tape files may also be suitable provided that they are in subfile format, and that the subfiles specified by the controlling parameters are confined to the following types:

1. Simple subfiles containing PLAN source only.
2. Simple subfiles containing semi-compiled segments only.
3. Composite subfiles containing any number of subfiles of types 1 and 2.
4. Simple subfiles containing consolidated semi-compiled program.

Any other types of subfile on the tapes are ignored by #XPMZ and #XPMY.

The type of subfile is indicated in word 12 of the start-of-subfile sentinel. For a discussion of composite tape file organization, and a description of the start-of-subfile sentinel, please refer to Chapter 10 of the manual 'Magnetic Tape'. The type of subfile indications in word 12 of the sentinels for the above four categories of subfile are B2P0, A300, C100 and A200 respectively.

PLAN source records may be variable length, in blocks with a maximum length of 210 words. The first record of each PLAN source segment must be a fixed length 21-word record containing a card image of a #STEER parameter; this must also be the first record in a block.

Semi-compiled records must be in batched blocks with a maximum length of 110 words. The output from all magnetic tape PLAN compilers is in this format except that if semi-compiled segments are input to #XPLN from a basic peripheral they are output in single record blocks. The result of trying to input semi-compiled segments in this format to #XPMX or #XPMZ is indeterminate.

**Editing Modes of #XPMX and #XPMZ**

A more detailed description of the two modes of operation of #XPMX and #XPMZ is given here.

**AMEND-IN—SITU**

When it is creating the output file, #XPMY takes a multiple of five buckets for each level 1 directory subfile; there may therefore be some spare space remaining unused at the end of each of these subfiles, distinct from the level 1 data subfile. When a program is subsequently updated by #XPMX or #XPMZ, amendments to it may include the addition of new segments. These require their subfile descriptions to be added to the level 1 directory subfile. If there is sufficient spare space in the directory subfile, the new subfile descriptions are added there, and the new subfiles for the additional segments are written utilizing some of the spare space which has hitherto
comprised the level 1 data subfile; the subfile description of the latter in the level 0 directory subfile is adjusted accordingly. If the existing level 1 directory subfile is full, however, the new subfile descriptions, as well as the new subfiles themselves, have to be written in the spare space which has hitherto comprised the level 1 data subfile. The level 1 directory subfile then becomes a fragmented subfile. Pointers are inserted in the directory description to indicate where fragments are to be found.

If segments are deleted, the subfiles themselves are not erased, but their subfile descriptions are removed from the level 1 directory subfile, so that the segments can no longer be accessed in the file. This has the effect of freeing more space in the level 1 directory subfile, so that subfile descriptions for new segments, that might otherwise have to go into the spare space of the level 1 data subfile, can go instead into the level 1 directory subfile. If a segment is replaced, the original segment is treated as for a deletion, the replacement segment goes into a new subfile, and the new subfile description takes the place of the old one in the level 1 directory subfile.

For each program in turn, the updating process is carried out in two phases. During phase one, parameters and amendments are read from paper tape and/or cards, and the parameters are printed on the line printer. Any subsidiary input files on magnetic tape and/or exchangeable disc are opened and copied as specified in the parameters. New subfiles are created for new or replacement segments, and their subfile descriptions are added at the end of the level 1 directory subfile. A system of temporary pointers inserted in the subfile descriptions indicates the desired sequence of segments and where the next subfile description in the sequence is to be found. Also during this phase amendments subfiles are created or updated as necessary; where the updating of an amendments subfile results in an increase in its size, the amendments subfile becomes fragmented. A work file is opened and is used in the process of updating amendments subfile fragments.

During phase two, the subfile descriptions in the level one directory subfile are reorganized, so as to get them into the required sequence and to eliminate redundant subfile descriptions and the temporary pointers. The work file is used again in this process. During this reorganization, a summary of the segments in the subfile is produced on the line printer. Dumps of the amendments subfiles of some or all of the source segments may be given on paper tape or cards if specified in the parameters. At the end of this phase, the data subfile description in the level 0 directory subfile is updated, and #XPMX or #XPMZ proceeds to update the next program.

COPY-AND-AMEND

In this mode, a fully reorganized copy of the updated file is written into a new file area specified by a parameter. For each program in the file, #XPMZ again works in two phases. During phase one, as the parameters and amendments are read, new segments are written into the new file area, and segments for which line amendments are specified are updated and written to the new file. When all the amendments have been read, segments to which no reference has been made in the amendments file are copied across, those with amendments subfiles being reorganized to produce updated source subfiles. The directory subfiles are copied onto the work file at the beginning of an update run, and are then treated in much the same way as in the amend-in-situ mode, temporary pointers being inserted in the level 1 subfile descriptions as the data subfiles are written to the new file area. When all the level 2 data subfiles have been written to the new file, the updating process enters its second phase, the level 1 directory subfile is reorganized and the updated directory subfiles are written to the new file. While this is taking place a summary of the new subfile is produced on the line printer, and dumps of some or all of the segments may be given on cards or paper tape and/or magnetic tape and/or the line printer. After all the subfiles referenced in the paper tape/card input file have been dealt with in this way, the subfiles of programs which have not been amended in this run are reorganized in a similar manner and written to the new file.

Input Files for the DISC COSY Compilers

The information in this sub-section is published for the benefit of those who may wish to develop their own routines for creating or editing source program files on disc. The files output by such routines may be submitted as input to the DISC COSY compilers if they comply with the specification given below. Any disc files which conform exactly to this specification will be acceptable as input to the appropriate compiler, #XPLZ or #XPLT; no guarantee is given, however, that such files are acceptable as input to #XPMX, #XPMY or #XPMZ.

This specification is not a definition of the format of files output by #XPMX, #XPMY or #XPMZ, nor is it a general specification for composite files on disc.

Files as specified below will contain only one program. They will otherwise have a similar structure to that illustrated for DISC COSY editor output files on page 45, but the level 2 data subfiles must each contain one segment in PLAN source form, with a steering record as the first record of the subfile; no subfile may contain semi-compiled program, unless it was copied from a file produced by the DISC COSY editors.

The bucket size is one block (128 words).
THE LEVEL 0 DIRECTORY SUBFILE

The Standard Bucket Header

The bucket comprising the level 0 directory subfile has a standard bucket header as follows:

<table>
<thead>
<tr>
<th>Word 0</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>Word 5</th>
<th>Word 6</th>
<th>Word 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 0 to 17</td>
<td>128 (in binary)</td>
<td>OFIL (Characters)</td>
<td>E (Characters)</td>
<td></td>
<td>Bits 0 to 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 18 to 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Zero</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Zero</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Record 0, the Directory Description

The record identifies the file as a composite file by using a standard name, 'COMPOFILE'. The contents of the record are as follows:

<table>
<thead>
<tr>
<th>Word 0</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>Word 5</th>
<th>Word 6</th>
<th>Word 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 0 to 17</td>
<td>COMP (Characters)</td>
<td></td>
<td></td>
<td></td>
<td>Bits 0 to 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 18 to 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Zero</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Zero</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Record 1, Subfile Description

This record describes the level 1 directory subfile. The contents of the record are as follows:

<table>
<thead>
<tr>
<th>Word 0</th>
<th>Word 1 to 3</th>
<th>Word 4</th>
<th>Word 5</th>
<th>Word 6</th>
<th>Word 7</th>
<th>Word 8</th>
<th>Word 9.</th>
<th>Word 10</th>
<th>Words 11 to 14</th>
<th>Word 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 0 to 17</td>
<td>Name of the level 1 directory subfile (characters).</td>
<td>Bits 0 to 5</td>
<td>Bit 0</td>
<td></td>
<td>Bits 0 to 22</td>
<td>CI100 (Characters)</td>
<td>Subfile generation number (in binary).</td>
<td>Date written (in binary).</td>
<td>Zero</td>
<td></td>
</tr>
<tr>
<td>Bits 18 to 23</td>
<td>Logical bucket number of the first bucket in the level 1 directory subfile (in binary).</td>
<td>Bits 6 to 23</td>
<td>Bits 1 to 5</td>
<td></td>
<td>Bit 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of buckets in the level 1 directory subfile (in binary).</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Record 2, Subfile Description

This record describes the data subfile comprising the spare space at the end of the file. The contents of the record are as follows:

Word 0  Bits 0 to 17     Zero
         Bits 18 to 23  16 (in binary)
Words 1 to 3
Word 4   Bits 0 to 5     Zero
         Bits 6 to 23  Logical bucket number of the first spare bucket (in binary).
Word 5   Bits 0 to 5     Zero
         Bits 6 to 23  Number of consecutive spare buckets (in binary).
Word 6
Word 7   Bits 0 to 22    Zero
         Bit 23        1
Words 8 to 15

THE LEVEL 1 DIRECTORY SUBFILE

The Standard Bucket Header

The buckets in this subfile have standard bucket headers of the same format as that described for the level 0 directory subfile.

Record 0, the Directory Description

The contents of this record are as follows:

Word 0  Bits 0 to 17     Zero
         Bits 18 to 23  8 (in binary)
Word 1 to 3
Word 4
Word 5   Bits 0 to 5     Zero
         Bits 6 to 23  Number of buckets in this subfile (in binary).
Word 6
Word 7   Bits 0 to 22    Zero
         Bit 23        1

Records 1 to n, Subfile Descriptions

These records describe the n level 2 data subfiles, each of which holds one PLAN source segment. The contents of each subfile description are as follows:

Word 0  Bits 0 to 17     Zero
         Bits 18 to 23  16 (in binary)
Words 1 to 3
Word 4   Bits 0 to 5     Zero
         Bits 6 to 23  Logical bucket number of the first bucket in the level 2 data subfile being described (in binary).
Word 5   Bits 0 to 5     Zero
         Bits 6 to 23  Number of buckets in the level 2 data subfile being described (in binary).
Word 6

Chapter 11
Word 7
Bits 0 to 5  Zero
Bits 6 to 23 Logical bucket number of the bucket which contains the level 0 directory subfile (in binary).

Word 8
Word 9
Word 10
Words 11 to 15

B2P0 (Characters)
Subfile generation number (in binary).
Date written (in binary).
Zero

LEVEL 2 DATA SUBFILES
The Standard Bucket Header
The buckets in these subfiles have standard bucket headers, of the same format as that described for the level 0 directory subfile.
Chapter 12  GEORGE 3 macros

INTRODUCTION
This chapter describes the GEORGE 3 System Macros available for running the compilers and other programs described in this manual. The macros described are:

<table>
<thead>
<tr>
<th>Macro name</th>
<th>Program(s)</th>
<th>Relevant chapter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>*PLAN</td>
<td>XPLF</td>
<td>7</td>
</tr>
<tr>
<td>PLAN 4</td>
<td>XPLN</td>
<td>8</td>
</tr>
<tr>
<td>PLAN4T</td>
<td>XPLT</td>
<td>8</td>
</tr>
<tr>
<td>*QPLAN</td>
<td>XPLF</td>
<td>7</td>
</tr>
<tr>
<td>QPLAN4</td>
<td>XPLN</td>
<td>8</td>
</tr>
<tr>
<td>*DACOSYCREATE</td>
<td>XPMY, XPLZ</td>
<td>7, 11</td>
</tr>
<tr>
<td>*DACOSYAMEND</td>
<td>XPMZ, XPLZ</td>
<td>7, 11</td>
</tr>
<tr>
<td>MTCOSYCREATE</td>
<td>XPMS, XPLV</td>
<td>7, 10</td>
</tr>
<tr>
<td>MTCOSYAMEND</td>
<td>XPMR, XPLV</td>
<td>7, 10</td>
</tr>
</tbody>
</table>

*The consolidation facilities in these macros involve the use of a free-standing consolidator.

PRESET STEERING FILES
Many of the macros described in this chapter use preset steering files. The contents of these are given below. The files which are used by particular macros may be determined from the lists of library entrants accessed given in the macro specifications.

All these steering files should be input as entrants in :LIB's directory and trapped to other users in READ mode. Those whose filename ends in PT should be punched to paper tape, the rest to cards. An extra newline or blank card (according to type) should be present to cater for the software double buffering. All punching should start in column one.

ICLSTXPLACD
ICLSTXPLAPT

OUTE(A)
NEXT(#XPCK)
BINE(B)
PEND
#SWITCH

ICLSTXPLBCD
ICLSTXPLBP7

OUTE(A)
NEXT(#XPCK)
BINE
PEND
#SWITCH
ICLSTXPCKA

*OUT ED(A)
*IN ED(B)
*LIB ED(C)
****

ICLSTXPCKB

*IN ED(B)
*LIB ED(C)
****

ICLSTXPLZ

IN(A)
OUT(B)
PEND

ICLSTXPMRCD

ICLSTXPRMT

IN(A)
OUT(B)
#SWITCH

ICLSTXPSACD

ICLSTXPSAPT

OUT(A)
IN(B)
COM(XPLV #TAPE)
#SWITCH

ICLSTXPSBCD

ICLSTXPSBPT

OUT(A)
COM(XPLV #TAPE)
#SWITCH

ICLSTXPLV

IN(A)
OUT(B)
PEND
MACRO PARAMETERS

In general the parameters for most of the macros described in this chapter follow the conventions described at the beginning of Chapter 13 of the ICL 1900 Series manual Operating Systems GEORGE 3 & 4, Edition 4 (TP4267). (The exceptions are the QPLAN and QPLAN4 macros whose parameters have fixed positions.) However note also that where a parameter may be used to specify an E.D.S. exofile or a magnetic tape there may be no spaces between the parameter identifying characters and the file description.

There are some parameter types which are common to many of the macros described (but not QPLAN and QPLAN4). These common parameter types will be described in the next section. In the remaining sections are descriptions of individual macros with details of the parameter types peculiar to them.

COMMON PARAMETER TYPES

The following parameter types are associated with the macros described (except QPLAN and QPLAN4):

  CHARACTER PERIPHERAL INPUT
  LISTING
  ERROR

Each parameter type is discussed under the following four headings:

1. POSITION If the parameter has a fixed position, this is stated. If it is optional or mandatory, this is stated. Note that if an optional parameter is omitted, no null parameter is necessary in its place.
2. FORMAT The format (including all permissible alternatives) is described.
3. FUNCTION The use of the parameter and its effect on the execution of the command is described.
4. NOTES Any additional information on the parameter types is given under this heading.

CHARACTER PERIPHERAL INPUT

Position

If there is only one Character Peripheral Input parameter, it must be the first parameter of the macro. If there is more than one, the first parameter of the macro must be a Character Peripheral Input parameter and the remaining parameters of this type may occupy any succeeding position(s). There must be at least one Character Peripheral Input parameter; otherwise an error will result.

Format

A Character Peripheral Input parameter consists of the characters *CR or *TR, optionally followed by a file description. If there is more than one parameter of this type, they need not have identical formats (see Function below). The format is:

  *CR file description
  *TR file description
  *CR
  or
  *TR

Function

A Character Peripheral Input parameter defines a file containing input (if the file description is included) or specifies that input is to be obtained from the job source (if the file description is omitted). The character peripheral input may be obtained partly from the job source and partly from a file by including the appropriate parameters.

Note

Any number of parameters of either peripheral type may be specified subject to the following restrictions:
The first parameter of the macro must be a Character Peripheral Input parameter. The program reading the
input will be entered at the entry point appropriate to the peripheral type.

All the parameters of one peripheral type must be written in the correct logical order.

If more than one parameter is specified, the last record to be read by the compiler for every parameter but
the last must contain an appropriate switching entry. Some compilers will, however, ignore a switching entry
if it calls for a peripheral of the same type; consequently the compiler will try to read off the end of the file.
The macro will detect such a condition and take appropriate action.

If the input is to be compiled partly from the job source, the part read from the job source must be compiled
first. In other words the first parameter must be *CR or *TR.

The total number of parameters in the macro must not exceed twenty-four.

Examples

1. A PLAN program is to be compiled from several files. These files are, in the order in which they are to be
   compiled:
      C1, C2, T3, C4, C5, T6, C7
   The last record of C2, T3, C5 and T6 must be
   #SWITCH
   The parameters for the card files must be in the order:
      C1, C2, C4, C5, C7
   and the parameters for the paper tape files in the order:
      T3, T6
   Thus the complete command could be written:
   PLAN *CR C1,*TR T3,*TR T6,*CR C2,*CR C4,*CR C5,*CR C7

2. The user has punched a PLAN program on paper tape, which has been input to the file SOURCE. To obtain
   a PLAN listing of it on his installation's line printer he can input the appropriate command and steering line
to a MOP console, thus:
   time ready PLAN *CR, *TRSOURCE
   ready #STEER LIST
   ready #SWITCH

LISTING

Position

The Listing parameter is optional and has no fixed position.

Format

A Listing parameter consists of the characters *LP optionally followed by a file description. The format is:

*LP file description

or

*LP

Function

A Listing parameter defines the file to which the line printer listing is to be written (if the file description is
included) or specifies that the listing is to be sent to the monitoring file system (if the file description is omitted)
in the OBJECT category. If the file specified in the file description already exists, the usual rules apply with regard
to overwriting it. It is not recommended that large amounts of output be sent to the monitoring file system.

Note that if more than 245K words are to be output, a multifile must be specified in the file description; a work
file cannot be used. If this parameter is omitted, the macro will use and erase a workfile.
ERROR
Position
The ERROR parameter is optional and has no fixed position.

Format
An Error parameter consists of the characters ER followed by a label. The format is:

    ER label

Function
An Error parameter defines a label to which control is to be passed in the event of a failure during the run or an error in the macro. If this parameter is omitted, the next command after the macro will be obeyed in the event of the run failing or an error being detected in the macro.

Notes
1. Special action is taken if a command error occurs in a macro issued in a MOP job (see Chapter 13 of the Operating Systems GEORGE 3 & 4 manual).
2. The label in the job description specified by the Error parameter must not coincide with any of the labels used within the macro, otherwise the command processor will go to the label within the macro when an error occurs. All macros that have an optional Error parameter have all their internal labels beginning with the character 9. Hence the user must not specify in the Error parameter any label beginning with 9.

SPECIFICATIONS OF SYSTEM MACROS
The remaining pages of this chapter are devoted to the specifications of system macros. The macros are described under the following headings:

1. FUNCTION  A brief description of the purpose of the macro is given here.
2. RESTRICTIONS  Any restrictions on the use of the macro, in addition to the usual contextual restrictions on the use of GEORGE 3 system macros described in Chapter 13 of the ICL 1900 Series manual Operating Systems GEORGE 3 & 4, are described here.
3. FORMAT  The format of each macro is given as the macro name followed by a parameter list. This parameter list consists of all the parameter types that may be used with the macro. The entries in the parameter list, that is, the various parameter types, are then listed and described under the same headings as were used for the description of Common Parameter Types. Where a parameter type has already been described under Common parameter types reference is made to that section and no further description is given.
4. EXECUTION  A brief description of the effect of the macro is given here.
5. LIBRARY ENTRANTS ACCESSED  A list of the library entrants accessed and the modes in which they are accessed is given under this heading.
6. ERROR MESSAGES  A list of references to chapters in this manual and to other ICL 1900 Series manuals containing explanations of the error messages that may be generated in the course of the execution of the macro is given. If the error messages are exclusive to the macro, they are explained.
PLAN

FUNCTION
The PLAN macro compiles a batch of one or more PLAN 3 programs. If the Binary parameter is given the first program in the batch with OBJECT in its steering line will also be consolidated.

RESTRICTIONS
Each source program in the batch must be preceded by a steering line, for example,

#STEER LIST,OBJECT

and the batch must be terminated by #STOP.

Normally the compiler parameters OUT, NEXT etc. are supplied from a preset steering file and should not be included in the source input. However the user has the option of supplying his own compiler parameters provided that he includes in his macro parameters the Own Parameter Indicator described below under Format. In that case the batch of source programs must be preceded by all the necessary #XPLF parameters described in Chapter 7, pages 27 to 32. However, where a file is already specified by a macro parameter, the filename given in the corresponding compiler parameter may be a dummy.

When using the Semi-compiled Library macro parameter described below under Format, the user must supply the compiler parameters. This is because in that case the compiler parameter LIBE must be included; and the subfile name specified in that compiler parameter must (unlike the filename, which may be a dummy) be the true name of the subfile to be used, which can be supplied only by the user.

A program with MAP or FULLIST specified in its steering line cannot be consolidated by this macro. However, such a program may be consolidated separately by the GEORGE 3 System Macro CONSOLIDATE. If the MAP or FULLIST facilities are to be implemented by the consolidator, then

1 The CONSOLIDATE macro should be called in the same job as the PLAN macro with no intervening core image and the first CONSOLIDATE macro parameter should be AUTO;

2 The semi-compiled version of the program must have been produced on a filestore file or an exofile to which the user still has WRITE access.

FORMAT

PLAN parameter list

Parameter types in the parameter list

CHARACTER PERIPHERAL INPUT
LISTING
ERROR

These parameter types are as described under Common parameter types. The other parameters in the parameter list are as follows:

COMPILER OUTPUT

Position

The Compiler Output parameter is optional and has no fixed position.

Format

A Compiler Output parameter consists of the characters COMP followed by an E.D.S. or F.D.S. file description or an E.D.S. exofile description. The format is:
COMP file description

or

COMPexofile description

Function
A Compiler Output parameter defines a disc file to which semi-compiled program is to be written. If a filestore
device is specified it must previously have been CREATED with one block buckets. The usual rules with regard to
overwriting a file apply. If this parameter is omitted, the macro will use and erase a workfile.

BINARY

Position
The Binary parameter is optional and has no fixed position.

Format
A Binary parameter consists of the characters BIN optionally followed by an E.D.S. or F.D.S. file description or
an E.D.S. exofile description. The format is:

BIN file description

or

BINexofile description

or

BIN

Function
A Binary parameter specifies that a binary object program is to be produced. It defines a disc file to which the
binary program is to be written (if the file or exofile description is included) or specifies that a core image of the
program is to be created (if the file description is omitted). If a filestore file is specified it must previously have
been CREATED with one block buckets. The usual rules with regard to overwriting a file apply. If this parameter
is omitted, no binary program will be produced.

Note
In the case of an overlay program, a file or exofile must always be specified.

SEMI-Compiled LIBRARY

Position
This parameter is optional and has no fixed position.

Format
This parameter consists of the characters SEMI followed by an E.D.S. or F.D.S. file description or an E.D.S.
exofile description. The format is:

SEMI file description

or

SEMIexofile description

Function
This parameter specifies a file or exofile containing semi-compiled routines which is to be ASSIGNED or ONLINEd
to the consolidator in addition to the Library file containing the standard PLAN subroutine group
:LIB.SUBGROUPS--RS. When this macro parameter is used, the user must supply the compiler parameters with
the source input as described above under Restrictions. A file or exofile specified by this macro parameter will be
opened and searched by the consolidator only if the corresponding compiler parameter LIBE is supplied. Note
that, even if the file specified by this macro parameter is an F.D.S. filestore file, the corresponding compiler
parameter must be LIBE (or LIB), not LIBF.
There may be up to six Semi-compiled Library parameters :LIB.SUBGROUPS-RS will always be assigned to the consolidator last after the files or exofiles specified by these macro parameters, but the number of files actually opened and searched by the consolidator is limited by the number of LIBE compiler parameters supplied with the source input. Therefore if :LIB.SUBGROUPS-RS is to be searched as well as the files or exofiles specified by the Semi-compiled Library parameters, the number of Semi-compiled Library parameters given must be one less than the number of LIBE parameters supplied with the source input. The largest number of LIBE parameters acceptable to the compiler is six, so, if the standard PLAN subroutine group is to be used, the maximum number of Semi-compiled Library parameters is five.

OWN PARAMETER INDICATOR

Position
This parameter is optional and has no fixed position.

Format
This parameter consists simply of the characters PAR.

Function
If present, this parameter indicates to the macro that the user has supplied all the necessary compiler parameters at the beginning of the source input stream. If it is omitted, the compiler parameters will be supplied from a preset steering file and any compiler parameters included in the source input will cause an error.

EXECUTION
The PLAN 3 compiler #XPLF is used to compile the source programs. If the Binary parameter is given, the System Macro CONSOLIDATE is called to consolidate the first program in the semi-compiled batch.

LIBRARY ENTRANTS ACCESSED

Entrant       Modes of access
:LIB.PROGRAM XPLF       EXECUTE, READ
:LIB.ICLTXPLFACD        READ
:LIB.ICLTXPLFAPT        READ
:LIB.ICLTXPLFBCD        READ
:LIB.ICLTXPLFBPT        READ
:LIB.PROGRAM XPCK       EXECUTE
:LIB.SUBGROUPS-RS        READ

The last two entrants are accessed by the System Macro CONSOLIDATE.

ERROR MESSAGES
Details of error messages produced by #XPLF may be found in Chapter 7. See also description of the System Macro CONSOLIDATE and the notes on consolidation in Chapter 13 of the ICL 1900 Series manual Operating Systems GEORGE 3 & 4, Edition 4 (TP4169).

Errors detected in the macro parameters and program error halts etc. occurring in the course of the macro will be reported by the general error message

DISPLAY : ERROR IN MACRO PLAN : MACRO ABANDONED.

The post-mortem procedure for compiler failures described on page 51 of Chapter 7 is included in this macro. A report of the contents of word 8 (if required) will be sent to the Monitoring File System but the post-mortem printout will be sent to a workfile and LISTFILEd on a line printer. Before starting the procedure the macro will DISPLAY : POSTMORTEM PROCEDURE STARTED and if during the procedure the Compiler again fails, the macro will DISPLAY : POSTMORTEM FAILED.
FUNCTION
The PLAN 4 system macro compiles and/or consolidates PLAN 4 programs using magnetic tape.

RESTRICTIONS
The user must supply all the character peripheral input required by the compiler #XPLN, control parameters as well as source and/or semi-compiled program.

When reading from paper tape #XPLN reads control parameters and source program in shift set mode but semi-compiled segments in graphic mode, each type of paper tape input must therefore be in files of the requisite mode, that is semi-compiled segments should not be in the same paper tape files as control parameters or source program.

Filestore magnetic tape files may not be used.

FORMAT
PLAN 4 parameter list

Parameter types in the parameter list
CHARACTER PERIPHERAL INPUT
LISTING
ERROR

These parameter types are described under Common parameter types.

EXECUTION
The PLAN 4 compiler #XPLN is loaded from a magnetic tape called PROGRAMTAPE and run. Filestore files may be used only to hold the character peripheral input and the compilation listing. Otherwise magnetic tapes are used. These are opened by unanticipated open mode PERIs issued by #XPLN in response to control parameters in the character peripheral input supplied by the user.

LIBRARY ENTRANTS ACCESSED
Entrant	Modes of Access
:LIB.PROGRAM XK62	EXECUTE
:LIB.PROGRAM TAPE	READ

This macro uses the FIND macro. #XPLN must be on the magnetic tape called PROGRAM TAPE and owned by :LIB.

ERROR MESSAGES
Details of error messages produced by #XPLN may be found in Chapter 8 of this manual.

Apart from the general error message ERROR IN MACRO PLAN4: MACRO ABANDONED no error messages are produced by the macro.
PLAN4T

FUNCTION
The PLAN4T macro compiles a batch of one or more PLAN 4 programs. If the Binary parameter is given the compiler's last output subfile with OBJECT in the program's steering line will also be consolidated. If the Binary parameter and the Fullist parameter are given, this program will also be full listed.

RESTRICTIONS
Each source program in the batch must be preceded by a steering line, for example,

#STEER LIST, OBJECT

or the compiler parameters must include a STEER parameter. The batch of programs should be preceded by all the necessary #XPLT parameters described in Chapter 8 of this manual, pages 18 to 27.

Where the Compiler Output parameter is specified, the file given in the Compiler Output parameter must be identical to the file specified in the OUTE compiler parameter. If this is a filestore file, it must be specified as a general local name and not as an absolute name (see manual Operating Systems GEORGE 3 and 4, page 48 for details of name format). It is not permissible to use a workfile (!) for this file.

Where a Binary parameter is specified, the filename given in the corresponding PLAN compiler parameter may be a dummy.

When a program is to be full listed, if a filename is not specified in the Binary parameter, the object program will be lost.

When using the Semicomplied Library macro parameters described below under Format, the user may use a dummy filename but the true name of the subfile should be used.

Where a Fullist parameter is specified, the subfile name in the compiler WSF parameter and the program name specified in the compiler PROG parameter for the program to be fulllisted should be identical to the name specified in the Fullist parameter.

FORMAT

PLAN4T parameter list

Parameter types in the parameter list

CHARACTER PERIPHERAL INPUT
LISTING
ERROR

These parameter types are as described under Common parameter types. Other parameters in the parameter list are described below.

COMPILER OUTPUT

Position
The Compiler Output parameter is optional and has no fixed position, but must be included if any program is to be consolidated.

Format
A Compiler Output parameter consists of the characters COMP followed by a DA file description or a DA exofile description. The format is:

COMP file description

or

COMP exofile description

10 Chapter 12 4322(7.72)
Function
A Compiler Output parameter defines a disc file from which semicompiled program is input to the consolidator. The same file should be specified as that used in the OUTE compiler parameter.

**BINARY**

**Position**
The Binary parameter is optional and has no fixed position.

**Format**
A Binary parameter consists of the characters BIN optionally followed by a DA file description or a DA exofile description. The format is:

- BIN file description
- BIN exofile description
- BIN

Function
A Binary parameter specifies that a binary object program is to be produced. It defines a disc file to which the binary program is to be written (if the file or exofile description is included) or specifies that a core image of the program is to be created (if the file description is omitted). If a filestore file is specified it must previously have been CREATED with one block buckets. The usual rules with regard to overwriting a file apply. If this parameter is omitted, no binary program will be produced.

**Note**
In the case of an overlay program, a file or exofile must always be specified.

**FULLLIST**

**Position**
The Fulllist parameter is optional and has no fixed position.

**Format**
A Fulllist parameter consists of the characters PROG followed by the four character name of the program to be fulllisted as specified in the compiler PROG parameter. The format is:

PROG name

Function
A Fulllist parameter indicates that the program whose name is specified in the parameter is to be fulllisted provided that a Binary parameter is also present.

**SEMICOMPILED LIBRARY**

**Position**
This parameter is optional and has no fixed position. It is applicable only when consolidation is required. One Semicomplied Library parameter must be supplied for each LIBE or INCE parameter included in the Character Peripheral Input except for the PLAN subroutine group SUBGROUPS-RS which will be searched automatically. The Semicomplied Library macro parameters should be in the same order as the corresponding LIBE or INCE parameters in the Character Peripheral Input. The LIBE parameter specifying SUBGROUPS-RS should be the last LIBE in the Character Peripheral Input.

**Format**
This parameter consists of the characters SEMI followed by a DA file description or a DA exofile description. The format is:
SEMIFiledescription

or

SEMExofiledescription

Function

Specifies a file or exofile containing semicomplied routines to be searched by the consolidator in addition to the file containing the standard PLAN subroutine group (:LIB.SUBGROUPS-RS).

EXECUTION

The PLAN 4 compiler #XPLT is used to compile the source programs. If the Binary parameter is given, the System Macro CONSOLIDATE is called to consolidate the last program in the semicomplied batch. The fullest program #XPSR is used to fullest the program that has been consolidated if a Fullist parameter is also given. The DA and MT input files for the compiler are specified in the compiler parameters and #XPLT issues appropriate ONLINE or ASSIGN commands for them.

LIBRARYENTRANTSACCESSED

<table>
<thead>
<tr>
<th>Entrant</th>
<th>Modes of access</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LIB.PROGRAM XPLT</td>
<td>EXECUTE,READ</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPSR</td>
<td>EXECUTE,READ</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPKC</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.SUBGROUPS-RS</td>
<td>READ</td>
</tr>
</tbody>
</table>

The last two entrants are accessed by the System Macro CONSOLIDATE.

ERRORMESSAGES

Details of error messages produced by #XPLT may be found in Chapter 8 of this manual, pages 29 and 30.

Details of error messages produced by #XPSR may be found in the manual PLAN Program Development Aids.

Details of error messages produced by the System Macro CONSOLIDATE may be found in the manual Operating Systems GEORGE 3 and 4, Chapter 13.

In addition the following two messages may be produced:

DISPLAY: YOUR PROGRAM IS TOO BIG FOR ENVIRONMENT

#XPCK has halted 'ST' indicating, in the case of a program to be consolidated into core, that the resulting binary program would exceed the maximum core size allowed. #XPCK will be deleted, the macro will be abandoned with the general error message mentioned below and control will be passed to the label specified in the error parameter if given. The job can be rerun after suitable adjustment of the appropriate installation parameter: alternatively the programmer must find some means of reducing the core size of his program.

DISPLAY: PLEASE ALLOCATE PERIPHERALS AND RESUME

#XPCK has halted CR, TR, LP, or TP. One of these halts will occur in the case of a program to be consolidated into core when #XPCK tries to implement the peripheral requests indicated by a #PERIPHERAL directive in the source program. The macro will return control to the user who should then ASSIGN or ONLINE all the peripherals requested by the #PERIPHERAL directive. If the command RESUME is then issued, the consolidator will move the program down to its correct place and then halt LD.

Errors detected in the macro parameters and program error halts occurring in the course of the macro will be reported by the general error message:

DISPLAY: ERROR IN MACRO PLAN4T: MACRO ABANDONED

The postmortem procedure for compiler failures is included in this macro. The postmortem printout will be sent to a workfile and LISTfiled on the line printer. Before starting the procedure the macro will

DISPLAY: POSTMORTEM PROCEDURE STARTED
and if during the procedure the compiler again fails, the macro will

DISPLAY: POSTMORTEM FAILED

If there is a failure while running the fulllist program #XPSR the message

DISPLAY: FULL LISTING ERROR

is output to the monitoring file.
QPLAN

FUNCTION
Provides a subset of the facilities offered by the System Macro PLAN. Compiles a batch of one or more PLAN 3 source programs and consolidates the first program in the batch with OBJECT in its steering line (if any).

RESTRICTIONS
All the character peripheral input is read as card input either from the job source or from a single filestore file.
Each program in the batch must be preceded by a steering line, for example

```
#STEER LIST,OBJECT
```

and the batch must be terminated by #STOP.
The compiler parameters, such as OUT, NEXT, should not be included.
A program whose steering line specifies MAP or FULLLIST cannot be consolidated.

FORMAT

QPLAN parameter list

Parameter types in the parameter list
Each parameter has a fixed position and is identified by that position. They are:

CHARACTER PERIPHERAL INPUT

Position
This parameter occupies the first parameter position.

Format
The parameter consists of the file description of a card image filestore file. The parameter may be omitted or null.

Function
The parameter indicates that the character peripheral input is to be read from the file specified. If omitted or null, it indicates that it is to be read from the job source.

BINARY PROGRAM

Position
This parameter occupies the second parameter position.

Format
The parameter consists of the file description of a previously created E.D.S. or F.D.S. filestore file to which the user has WRITE access. The parameter may be omitted or null.

Function
The parameter indicates that the consolidated program is to be left in the file specified. If omitted or null, it indicates that the first program in the compiled batch is to be consolidated into core.

LISTING

Position
This parameter occupies the third parameter position.
Format
The parameter consists of any character or set of characters excluding - (hyphen) and % (percent). The parameter may be omitted or null.

Function
If present a listing parameter specifies that the compilation and consolidation listings are to be output to the monitoring file and job source at the end of the macro, as well as being listed on the line printer. If it is omitted or null, the macro will use and erase a workfile, listing it on the line printer.

EXECUTION
The macro loads and runs the PLAN 3 compiler #XPLF, and, if there is a program to be consolidated, the consolidator #XPCK.

LIBRARY ENTRANTS ACCESSED

<table>
<thead>
<tr>
<th>Entrants</th>
<th>Modes of access</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LIB.PROGRAM XPLF</td>
<td>EXECUTE, READ</td>
</tr>
<tr>
<td>:LIB.ICLSTXPLFAPT</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.ICLSTXPLFBPT</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPCK</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.SUBGROUPS-RS</td>
<td>READ</td>
</tr>
</tbody>
</table>

ERROR MESSAGES
Details of error messages produced by #XPLF and #XPCK may be found in Chapter 7 of this manual and Chapter 6 of the 1900 Series manual Compiling Systems respectively.

No error messages will be produced by this macro, however, the following message will be produced if #XPCK halts CR, TR, LP, CP or TP:

DISPLAY : PLEASE ALLOCATE PERIPHERALS AND RESUME

One of these halts will occur in the case of a program to be consolidated into core when #XPCK tries to implement the peripheral requests indicated by a #PERIPHERAL directive in the source program. The macro will return control to the user who should then ASSIGN or ONLINE all the peripherals requested by the #PERIPHERAL directive. If the command RESUME is then issued, the consolidator will move the program down to its correct place and then halt LD.

4322(7.72)  Chapter 12  15
QPLAN4

FUNCTION
Provides a subset of the facilities offered by the System Macro PLAN 4.

RESTRICTIONS
All the character peripheral input is read as card input either from the job source or from a single filestore file.
The user must supply all the character peripheral input required by the compiler #XPLN, control parameters as well as source and/or semi-compiled program.
Filestore magnetic tape files may not be used.

FORMAT

QPLAN 4 parameter list

Parameter types in the parameter list
Each parameter has a fixed position and is identified by that position.

CHARACTER PERIPHERAL INPUT

Position
This parameter occupies the first parameter position.

Format
The parameter consists of the file description of a card image filestore file. The parameter may be omitted or null.

Function
The parameter indicates that the character peripheral input is to be read from the file specified. If omitted or null, it indicates that it is to be read from the job source.

LISTING

Position
This parameter occupies the third parameter position. Therefore, a null parameter should be given in the second parameter position if the Listing parameter is present.

Format
This parameter consists of any character or set of characters excluding – (hyphen) and % (percent). The parameter may be omitted or null.

Function
If present a listing parameter specifies that the compilation and consolidation listings are to be output to the monitoring file and job source at the end of the macro, as well as being listed on the line printer. If it is omitted or null, the macro will use and erase a workfile, listing it on the line printer.

EXECUTION
The PLAN 4 compiler #XPLN is loaded from a magnetic tape called PROGRAM TAPE and run. Filestore files may be used only to hold the character peripheral input. Otherwise magnetic tapes are used. These are opened by unanticipated open mode PERIs issued by #XPLN in response to control parameters in the character peripheral input supplied by the user.
LIBRARY ENTRANTS ACCESSED

Entrant                        Modes of access
:LIB.PROGRAM XK62               EXECUTE
:LIB.PROGRAM TAPE               READ

This macro uses the FIND macro. #XPLN must be on the magnetic tape called PROGRAM TAPE and owned by :LIB.

ERROR MESSAGES

Details of error messages produced by #XPLN may be found in Chapter 8 of this manual. No error messages are produced by the macro itself.
DACOSYCREATE

FUNCTION
The DACOSYCREATE macro is used to create a COSY file. Unless the COSY control parameters specify a 'create only' run, the programs in the file which have the appropriate steering lines will be compiled by a PLAN 3 compiler and, if the appropriate parameters are given, the first program in the compiled batch will be consolidated.

RESTRICTIONS
The user must supply as character peripheral input all the control parameters amendments to be read by XPMY. In particular he must supply the initial control parameters OUTE or OUTF, APPE or APPF, OUTW and the compiler parameter NEXT (#XPLZ). The IN parameters associated with any #FILE parameters must be included in the input. The user is not required to supply separate character peripheral input for the compiler or the consolidator. If compilation and/or consolidation is required the character peripheral input should be supplied to XPMY by means of the #CHAIN parameter described on page 8 of Chapter 11.

Exofile and magnetic tape descriptions specified in the macro parameters must include storage unit or tape serial numbers. Magnetic tape descriptions are not permitted to include the character ? in place of serial numbers. The names and generation number of filestore files will not be changed.

When consolidation of the first program in the compiled batch is specified (by the binary parameter) the consolidator will not implement a steering line of MAP or FULLIST for that program.

FORMAT
DACOSYCREATE parameter list

Parameter types in the parameter list
CHARACTER PERIPHERAL INPUT
LISTING
ERROR

These parameter types are as described under Common parameter types. Other parameters in the parameter list are described below.

BINARY
Position
The binary parameter is optional and has no fixed position.

Format
A binary parameter consists of the characters BIN optionally followed by a DA file description or exofile description. The format is:

BIN file description

or

BIN exofile description

Function
A Binary parameter defines a direct access file to which the binary program is to be written. If a filestore file is specified it must previously have been created with one block buckets. The usual rules with regard to overwriting a file apply.

This parameter should only be supplied if the PLAN compiler parameter BINE or BINF is included in the character peripheral input and is accompanied by a filename.
If this parameter is omitted and the PLAN compiler parameter BINE or BINF is included in the character peripheral input without a filename, the binary program will be left in core.

Note
In the case of an overlay program, an output file must always be specified.

SEMICONPILED LIBRARY

Position
This parameter has no fixed position and is optional. It is applicable only when consolidation is required. One semi-compiled library parameter must be supplied for each LIBE or LIBF parameter included in the character peripheral input except for the PLAN subroutine group SUBGROUPS-RS which will be searched automatically. The semiconpiled library macro parameters should be in the same order as the corresponding LIBE or LIBF parameters in the character peripheral input. The LIBE or LIBF parameter specifying SUBGROUPS-RS should be the last LIBE or LIBF in the character peripheral output.

Format
This parameter consists of the characters SEMI followed by a DA file description or exofile description. The format is:

SEMI file description
or
SEMI exofile description

Function
Specifies a file or exofile containing semiconpiled routines to be searched by the consolidator in addition to the file containing the standard PLAN subroutine group (:LIB.SUBGROUPS-RS).

Note
Although the filenames specified in the LIBE or LIBF parameters in the character peripheral input may be dummy names (since they are overridden by the file descriptions specified in the semiconpiled library parameters supplied to the macro) the subfile names should be the true names of the subfiles to be searched.

INPUT AND OUTPUT

Position
The input and output parameters have no fixed position.

Format
The input and output parameters consist of a dummy filename corresponding to a filename given in the program control parameters in the character peripheral input, followed by the fixed character = and a DA or MT file description, or a DA exofile description or a magnetic tape description. A dummy filename may consist of any string of characters acceptable to the COSY or PLAN programs, though character strings which could be confused with other parameter identifiers (for example, BIN) should be avoided. The format is:

dummy filename = file description
or
dummy filename = exofile description
or
dummy filename = magnetic tape description

Function
Defines a direct access or magnetic tape filestore file, or a direct access exofile or a magnetic tape associated with the following COSY and PLAN compiler file parameters:
COSY OUTE or OUTF
APPE or APPF
OUTW
IN associated with #M FILE parameters

PLAN OUTE or OUTF or APPE or APPF

These control parameters are described fully in Chapters 7 and 11 of this manual.

NOTES

1. A macro parameter should appear for each of the required COSY or PLAN control parameters in the character peripheral input described above unless the required files are to be opened by unanticipated open-mode PERI instructions (see note 4 below).

2. The dummy filenames in the program control parameters should not be accompanied by storage unit or tape serial numbers, but may include file generation numbers or reel sequence numbers, although these details will be ignored.

3. The COSY output file will be used as the only input file for the PLAN compiler, therefore the filename given in the INE control parameter to the PLAN compiler will be ignored although it must be present.

4. If exofiles or magnetic tapes are used rather than filestore files, and no input or output parameters are supplied to the macro, the required files will be opened by unanticipated open-mode PERI instructions, with the exception of the PLAN compiler output file. If it is required to write the compiler output to a named filestore file or exofile, a macro parameter of the form described above must be supplied. If such a parameter is not supplied the macro will use and erase a workfile.

5. Use of the COSY control parameters RENE or RENF and the PLAN control parameter REN will have the following effects:
   (a) If the file concerned is a filestore file the parameter will have no effect.
   (b) If the file is an exofile and is the COSY output file and compilation is required, the PLAN compiler will open that file by unanticipated open-mode PERI instruction even if the old exofile name was specified in a macro parameter.

   Note: The PLAN control parameter REN should not be used to rename an exofile if consolidation is required, as the old name of the exofile will be passed to the CONSOLIDATE macro and an error will result.
   (c) If the COSY control parameter OUTW is used in the form

   OUTW (####, serial number)

   a workfile will be used and erased.

EXECUTION

The macro loads and runs #XPMY and possibly #XPLZ and #XPCK.

Library entrants accessed

<table>
<thead>
<tr>
<th>Entrant</th>
<th>Modes of access</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LIB.PROGRAM XPMY</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPLZ</td>
<td>READ,EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPCK</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.SUBGROUPS:RS</td>
<td>READ</td>
</tr>
</tbody>
</table>

The last two entrants are accessed by the system macro CONSOLIDATE.
ERROR MESSAGES

Details of error messages produced by #XPMY and #XPLZ may be found in Chapters 11 and 7 of this manual respectively; the error messages produced by #XPCK are described in 1900 Series Compiling Systems (First edition, TP 4241) Chapter 6.

If a PLAN compiler output parameter is not included in the character peripheral input, the following message will be sent to the monitoring file and the macro will be abandoned:

NO SEMICOMPiled OUTPUT PRODUCED BY XPLZ

The postmortem procedure, described on page 51 of Chapter 7 of this manual, is entered if a failure is detected in the course of compilation. A report of the contents of Word 8 (if required) will be sent to the monitoring file system but the postmortem printout will be sent to a workfile and LISTFILEd on a line printer. Before starting the procedure the macro will DISPLAY: POSTMORTEM PROCEDURE STARTED and if during the procedure the compiler again fails, the macro will DISPLAY: POSTMORTEM FAILED.

Details of error messages produced by the system macro CONSOLIDATE may be found in Operating Systems GEORGE 3 and 4, Chapter 13.

In addition the following two messages may be produced;

DISPLAY: YOUR PROGRAM IS TOO BIG FOR ENVIRONMENT

#XPCK has halted 'ST' indicating, in the case of a program to be consolidated into core, that the resultant binary program would exceed the maximum core size allowed. #XPCK will be deleted, the macro will be abandoned with the general error message mentioned above and control will be passed to the label specified in the error parameter, if given. The job can be rerun after suitable adjustment of the appropriate installation parameter; alternatively the programmer must find some means of reducing the core size of his program.

DISPLAY: PLEASE ALLOCATE PERIPHERALS AND RESUME

#XPCK has halted CR, TR, LP or TP. One of these halted will occur in the case of a program to be consolidated into core when #XPCK tries to implement the peripheral requests indicated by a #PERIPHERAL directive in the source program. The macro will return control to the user who should then ASSIGN or ONLINE all the peripherals requested by the #PERIPHERAL directive. If the command RESUME is then issued, the consolidator will move the program down to its correct place and then halt LD.

All errors occurring during the course of the macro will be reported by the general error message

ERROR IN DACOSYCREATE: MACRO ABANDONED
DACOSYAMEND

FUNCTION

The DACOSYAMEND macro is used to update a batch of PLAN source programs in disc COSY format. Unless the COSY control parameters specify an 'update only' run, the program in the batch which have the appropriate steering lines will be compiled by a PLAN 3 compiler, and if the appropriate parameters are given, the first program in the compiled batch will be consolidated.

RESTRICTIONS

The user must supply as character peripheral input all the control parameters and amendments to be read by #XPMZ. In particular he must supply the initial control parameters INE or INF, OUTE or OUTF, OUTW, and the compiler parameter NEXT (#XPLZ). The initial control parameter OUT or APP must be supplied if programs or segments are to be copied to magnetic tape, and the IN, INE or INF parameters associated with any #MFILE or #DFILE parameters included in the input. The user is not required to supply separate character peripheral input for the compiler or the consolidator. If compilation and/or consolidation is required the character peripheral input should be supplied to #XPMZ by means of the #CHAIN parameter described on page 29 of Chapter 11.

Exoffile and magnetic tape descriptions specified in the macro parameters must include storage unit or tape serial numbers. Magnetic tape descriptions are not permitted to include the character ? in place of serial numbers. The names and generation numbers of filestore files will not be changed.

When consolidation of the first program in the compiled batch is specified (by the binary parameter) the consolidator will not implement a steering line of MAP or FULLIST for that program.

FORMAT

DACOSYAMEND parameter list

Parameter types in the parameter list

CHARACTER PERIPHERAL INPUT
LISTING
ERROR

These parameter types are as described under Common parameter types. Other parameters in the parameter list are described below.

BINARY

Position

The binary parameter is optional and has no fixed position.

Format

A binary parameter consists of the characters BIN optionally followed by a DA file description or exofile description. The format is:

BIN file description

or

BIN exofile description

Function

A binary parameter defines a direct access file to which the binary program is to be written. If a filestore file is specified it must previously have been created with one block buckets. The usual rules with regard to overwriting a file apply.
This parameter should only be supplied if the PLAN compiler parameter BINE or BINF is included in the character peripheral input and is accompanied by a filename.

If this parameter is omitted and the PLAN compiler parameter BINE or BINF is included in the character peripheral input without a filename, the binary program will be left in core.

Note
In the case of an overlay program, an output file must always be specified.

SEMICONPILED LIBRARY

Position
This parameter has no fixed position and is optional. It is applicable only when consolidation is required. One semicompiled library parameter must be supplied for each LIBE or LIBF parameter included in the character peripheral input except for the PLAN subroutine group SUBGROUPS-RS which will be searched automatically. The semicompiled library macro parameters should be in the same order as the corresponding LIBE or LIBF parameters in the character peripheral input. The LIBE or LIBF parameter specifying SUBGROUPS-RS should be the last LIBE or LIBF in the character peripheral input.

Format
This parameter consists of the characters SEMI followed by a DA file description or exofile description. The format is:

SEMI file description

or

SEMI exofile description

Function
Specifies a file or exofile containing semicompiled routines to be searched by the consolidator in addition to the file containing the standard PLAN subroutine group (:LIB.SUBGROUPS-RS).

Note
Although the filenames specified in the LIBE or LIBF parameters in the character peripheral input may be dummy names (since they are overridden by the file description specified in the semicompiled library parameters supplied to the macro) the subfile names should be the true names of the subfiles to be searched.

INPUT AND OUTPUT

Position
The input and output parameters have no fixed position.

Format
The input and output parameters consist of a dummy filename corresponding to a filename given in the program control parameters in the character peripheral input, followed by the fixed character = and a DA or MT file description, or a DA exofile description or a magnetic tape description. A dummy filename may consist of any string of characters acceptable to the COSY or PLAN programs, though character strings which could be confused with other parameter identifiers (for example, BIN) should be avoided. The format is:

dummy filename = file description

or

dummy filename = exofile description

or

dummy filename = magnetic tape description

Function
Defines a direct access or magnetic tape filestore file or a direct access exofile or a magnetic tape associated with the following COSY and PLAN compiler file parameters:
COSY INE or INF
OUTE or OUTF
OUT or APP
OUTW
INE associated with #DFILE parameters
IN associated with #MFILE parameters

PLAN OUTE or OUTF or APPE or APPF

These control parameters are described fully in Chapters 7 and 11 of this manual.

NOTES

1 A macro parameter should appear for each of the required COSY or PLAN control parameters in the
character peripheral input described above unless the required files are to be opened by unanticipated
open-mode PERI instructions (see note 4 below).

2 The dummy filenames in the program control parameters should not be accompanied by storage unit or
tape serial numbers, but may include file generation numbers or reel sequence numbers, although these
details will be ignored.

3 The COSY output file will be used as the only input file for the PLAN compiler, therefore the filename
given in the INE control parameter to the PLAN compiler will be ignored although it must be present.

4 If exofiles or magnetic tapes are used rather than file-store files, and no input or output parameters are
supplied to the macro, the required files will be opened by unanticipated open-mode PERI instructions,
with the exception of the PLAN compiler output file. If it is required to write the compiler output to a
named file-store file or exofile, a macro parameter of the form described above must be supplied. If such a
parameter is not supplied the macro will use and erase a workfile.

5 If a dummy filename is given in the COSY control parameters INE, INF, OUTE or OUTF in the character
peripheral input, this dummy name will appear in the COSY line printer output.

6 Use of the COSY control parameters RENE or RENF and the PLAN control parameter REN will have the
following effects:

(a) If the file concerned is a filestore file the parameter will have no effect except that after a RENE or
RENF parameter the new name will appear on the COSY line printer listing.

(b) If the file is an exofile and is the COSY output file and compilation is required, the PLAN compiler
will open that file by unanticipated open-mode PERI instruction even if the old exofile name was
specified in a macro parameter.

Note: The PLAN control parameter REN should not be used to rename an exofile if consolidation
is required as the old name of the exofile will be passed to the CONSOLIDATE macro and an error
will result.

(c) The COSY control parameter REN (for magnetic tape) will cause a magnetic tape to be renamed. If a
filestore file is concerned, the effect will be as described on page 475 of Operating Systems GEORGE
3 and 4 for mode #500 PERIs, and the filestore filename will not be changed.

(d) If the COSY control parameter OUT or APP is used in the form

OUT (#####)

a pool tape will be obtained and renamed SCRATCH TAPE. This may then be followed by a REN
parameter to relabel the tape. The form

OUT (#####, serial no)

should not be used.

(e) If the COSY control parameter OUTW is used in the form

OUTW (#####, serial number)

a workfile will be used and erased.
EXECUTION

The macro loads and runs #XPMZ and possibly #XPLZ and #XPCK. Paper tape and card output from #XPMZ will be sent to a work file which will be LISTFILEd to the appropriate peripheral and erased.

Library entrances accessed

<table>
<thead>
<tr>
<th>Entrance</th>
<th>Modes of access</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LIB.PROGRAM XPMZ</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPLZ</td>
<td>READ, EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM XPCK</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.SUBGROUPS-RS</td>
<td>READ</td>
</tr>
</tbody>
</table>

The last two entrances are accessed by the system macro CONSOLIDATE.

ERROR MESSAGES

Details of error messages produced by #XPMZ and #XPLZ may be found in Chapters 11 and 7 of this manual respectively. The error messages produced by #XPCK are described in 1900 Series Compiling Systems (First edition, TP 4241), Chapter 6.

If a PLAN compiler output parameter is not included in the character peripheral input, the following message will be sent to the monitoring file and the macro will be abandoned:

NO SEMICOMPiled OUTPUT PRODUCED BY XPLZ

The postmortem procedure, described on page 51 of Chapter 7 of this manual, is entered if a failure is detected in the course of compilation. A report of the contents of Word 8 (if required) will be sent to the monitoring file system but the postmortem printout will be sent to a work file and LISTFILEd on a line printer. Before starting the procedure the macro will DISPLAY: POSTMORTEM PROCEDURE STARTED and if during the procedure the compiler again fails, the macro will DISPLAY: POSTMORTEM FAILED.

Details of error messages produced by the system macro CONSOLIDATE may be found in Operating Systems GEORGE 3 and 4, Chapter 13.

In addition the following two messages may be produced:

DISPLAY: PLEASE ALLOCATE PERIPHERALS AND RESUME

#XPCK has halted 'ST' indicating, in the case of a program to be consolidated into core, that the resultant binary program would exceed the maximum core size allowed. #XPCK will be deleted, the macro will be abandoned with the general error message mentioned above and control will be passed to the label specified in the error parameter, if given. The job can be rerun after suitable adjustment of the appropriate installation parameter; alternatively the programmer must find some means of reducing the core size of his program.

DISPLAY: PLEASE ALLOCATE PERIPHERALS AND RESUME

#XPCK has halted CR, TR, CP or TP. One of these halts will occur in the case of a program to be consolidated into core when #XPCK tries to implement the peripheral requests indicated by a #PERIPHERAL directive in the source program. The macro will return control to the user who should then ASSIGN or ONLINE all the peripherals requested by the #PERIPHERAL directive. If the command RESUME is then issued, the consolidator will move the program down to its correct place and then halt LD.

All errors occurring during the course of the macro will be reported by the general error message:

ERROR IN DACOSYAMEND: MACRO ABANDONED
MT cosycreate

FUNCTION
The MT cosycreate macro may be used to create a COSY file on magnetic tape or in a filestore magnetic tape file. If the Binary parameter is given those programs in the file which have the appropriate steering line will be compiled.

RESTRICTIONS
The user must not include in the character peripheral input the initial control parameters for #XPMS (see Chapter 10, page 4) though he must supply all the other control parameters and amendments to be read by that program. In particular he must supply the IN parameters associated with any #FILE parameters included in the input, although where the file or tape to be used is specified by the macro parameter Additional Magnetic Tape Input described below, the name given in the IN parameter read by #XPMS may be a dummy. The user is not required to supply character peripheral input for the compiler.

A magnetic tape description must be enclosed in parenthesis, that is, it must be either:

(serial number)

or

(serial number, tape name)

If compiling, MT cosycreate uses the FIND macro. The compiler #XPLV must be located on the magnetic tape called PROGRAM TAPE and owned by :LIB.

FORMAT
MT cosycreate parameter list

Parameter types in the parameter list
- CHARACTER PERIPHERAL INPUT
- LISTING
- ERROR

These parameter types are as described under Common parameter types. Other parameters in the parameter list are described below.

COSY FILE

Position
The COSY File parameter has no fixed position but must be included in the parameter list.

Format
The COSY File parameter consists of the characters COSY followed by a file description. The format is:

COSY magnetic tape description

or

COSY filestore MT file description

Function
The COSY File parameter defines the magnetic tape file which is to hold the COSY file. If the parameter is omitted, an error will occur.
MAGNETIC TAPE INPUT

Position
This parameter has no fixed position and is optional.

Format
This parameter consists of the characters FILE followed by a file description. The format is:

FILE  magnetic tape description

or

FILE  filestore MT file description

Function
This parameter defines a file from which a complete COSY file is to be filed across. If the parameter is omitted, no such filing across is possible.

ADDITIONAL MAGNETIC TAPE INPUT

Position
This parameter has no fixed position and is optional.

Format
This parameter consists of the characters SEGS followed by a file description. The format is:

SEGS  magnetic tape description

or

SEGS  filestore MT file description

Function
This parameter defines the file from which segments are to be filed (see Chapter 10, page 9). Only one such parameter can be used; any other files must be magnetic tapes and must be opened by unanticipated open mode PERI.

BINARY

Position
This parameter has no fixed position and is optional.

Format
This parameter consists of the characters BIN followed by a file description. The format is:

BIN  magnetic tape description

or

BIN  filestore MT file description

Function
This parameter defines a file to hold binary output from #XPLV. If the parameter is omitted, no compilation occurs.

EXECUTION
The MTCOSYCREATE macro loads and runs #XPMS and possibly #XPLV.
LIBRARY ENTRANTS ACCESSED

<table>
<thead>
<tr>
<th>Entrant</th>
<th>Modes of access</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LIB.PROGRAM XPMS</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM XK62</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>:LIB.PROGRAM TAPE</td>
<td>READ (see above under Restrictions)</td>
</tr>
<tr>
<td>:LIB.ICALSTXPMSAPT</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.ICALSTXPMSBPT</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.ICALSTXPMSACD</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.ICALSTXPMSBCD</td>
<td>READ</td>
</tr>
<tr>
<td>:LIB.ICALSTXPLV</td>
<td>READ</td>
</tr>
</tbody>
</table>

ERROR MESSAGES

Details of error messages produced by #XPMS and #XPLV may be found in Chapters 10 and 7.
MTCOSYAMEND

FUNCTION

The MTCOSYAMEND macro may be used to update COSY files held on magnetic tape or in filestore magnetic tape files. If the Binary parameter is given those programs in the file which have the appropriate steering lines will be compiled.

RESTRICTIONS

The user must not include in the character peripheral input the initial control parameters for #XPMR (see Chapter 10, page 20) though he must supply all the other control parameters and amendments to be read by that program. In particular he must supply the INA parameters associated with any #FILE parameters included in the input, although where the file or tape to be used is specified by the macro parameter Additional Magnetic Tape Input described below, the name given in the INA parameter read by #XPMR may be a dummy. The user is not required to supply character peripheral input for the compiler.

A magnetic tape description must be enclosed in parenthesis, that is, it must be either:

(serial number)

or

(serial number, tape name)

If compiling, MTCOSYAMEND uses the FIND macro. The compiler #XPLV must be located on the magnetic tape called PROGRAM TAPE and owned by :LIB.

FORMAT

MTCOSYAMEND  parameter list

Parameter types in the parameter list

CHARACTER PERIPHERAL INPUT
LISTING
ERROR

The parameter types are as described under Common parameter types. Other parameters in the parameter list are described below.

INPUT COSY FILE

Position

The Input COSY File parameter has no fixed position but must be given in the parameter list.

Format

The Input COSY File parameter consists of the characters COSYIN followed by a file description. The format is:

COSYIN  magnetic tape description

or

COSYIN  filestore MT file description

Function

The Input COSY File parameter defines the file which holds the brought forward COSY file. If this parameter is omitted, an error will occur.
OUTPUT COSY FILE

Position
The Output COSY File parameter has no fixed position but must be included somewhere in the parameter list.

Format
The Output COSY File parameter consists of the characters COSYOUT followed by a file description. The format is:

COSYOUT magnetic tape description

or

COSYOUT filestore MT file description

Function
The Output COSY File parameter defines the file which is to hold the carried forward COSY file. If this parameter is omitted, an error will occur.

ADDITIONAL MAGNETIC TAPE INPUT

Position
This parameter has no fixed position and is optional.

Format
This parameter consists of the characters SEGS followed by a file description. The format is:

SEGS magnetic tape description

or

SEGS filestore MT file description

Function
This parameter defines a file from which segments are to be filed (see Chapter 10, page 32). Only one such parameter can be used; any other files must be magnetic tapes and must be opened by an unanticipated open mode PERI.

BINARY

Position
This parameter has no fixed position and is optional.

Format
This parameter consists of the characters BIN followed by a file description. The format is:

BIN magnetic tape description

or

BIN filestore MT file description

Function
This parameter defines a file which is to hold binary output from #XPLV. If this parameter is omitted, no compilation will occur.

EXECUTION
The MT COSY AMEND macro loads and runs #XPMR and, if a Binary parameter is specified, #XPLV is also loaded and run.
LIBRARY ENTRANTS ACCESSED

Entrant                                      Modes of access
:LIB.PROGRAM XPMR                           EXECUTE
:LIB.PROGRAM XK62                           EXECUTE
:LIB.PROGRAM TAPE                           READ (see above under Restrictions)
:LIB.ICLSTXPMPRT                             READ
:LIB.ICLSTXPMRCD                             READ
:LIB.ICLSTXPLV                               READ

ERROR MESSAGES
Details of error messages produced by #XPMR and #XPLV may be found in Chapters 10 and 7.
PLAN REFERENCE MANUAL PART 4

Appendices
# Appendix I  ICL 1900 Series Codes

## The 1900 Series 64 – Character Card Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Card Punching</th>
<th>Symbol</th>
<th>Card Punching</th>
<th>Symbol</th>
<th>Card Punching</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>F</td>
<td>10/6</td>
<td>-</td>
<td>(minus/hyphen)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>G</td>
<td>10/7</td>
<td>&quot;</td>
<td>(quotes)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>H</td>
<td>10/8</td>
<td>/</td>
<td>(solidus)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>I</td>
<td>10/9</td>
<td>+</td>
<td>(plus)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>J</td>
<td>11/1</td>
<td>.</td>
<td>(stop)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>K</td>
<td>11/2</td>
<td>;</td>
<td>(semi-colon)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>L</td>
<td>11/3</td>
<td>:</td>
<td>(colon)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>M</td>
<td>11/4</td>
<td>'</td>
<td>(apostrophe)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>N</td>
<td>11/5</td>
<td>!</td>
<td>(exclamation)</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>O</td>
<td>11/6</td>
<td>[</td>
<td>(left bracket)</td>
</tr>
<tr>
<td>Space</td>
<td>NONE</td>
<td>P</td>
<td>11/7</td>
<td>$</td>
<td>(dollar)</td>
</tr>
<tr>
<td>&amp; (ampersand)</td>
<td>10 or 10/0</td>
<td>Q</td>
<td>11/8</td>
<td>*</td>
<td>(asterisk)</td>
</tr>
<tr>
<td># (number)</td>
<td>3/8</td>
<td>R</td>
<td>11/9</td>
<td>&gt;</td>
<td>(greater than)</td>
</tr>
<tr>
<td>@</td>
<td>4/8</td>
<td>S</td>
<td>0/2</td>
<td>&lt;</td>
<td>(less than)</td>
</tr>
<tr>
<td>( (left parenthesis)</td>
<td>5/8</td>
<td>T</td>
<td>0/3</td>
<td>†</td>
<td></td>
</tr>
<tr>
<td>) (right parenthesis)</td>
<td>6/8</td>
<td>U</td>
<td>0/4</td>
<td>£</td>
<td>(pound)</td>
</tr>
<tr>
<td>] (right bracket)</td>
<td>7/8</td>
<td>V</td>
<td>0/5</td>
<td>;</td>
<td>(comma)</td>
</tr>
<tr>
<td>A</td>
<td>10/1</td>
<td>W</td>
<td>0/6</td>
<td>%</td>
<td>(percentage)</td>
</tr>
<tr>
<td>B</td>
<td>10/2</td>
<td>X</td>
<td>0/7</td>
<td>?</td>
<td>(question)</td>
</tr>
<tr>
<td>C</td>
<td>10/3</td>
<td>Y</td>
<td>0/8</td>
<td>=</td>
<td>(equals)</td>
</tr>
<tr>
<td>D</td>
<td>10/4</td>
<td>Z</td>
<td>0/9</td>
<td>←</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>10/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The 8-Track (7 Data Bit) 1900 Paper Tape Code

The 8-track paper tape code employs 7 data bits and 1 parity bit. Of the 7 data bits, three are used for zone identification (0-7) and 4 for numeric identification 0-15. The following three examples illustrate the use of the two groups of bits. A punching is taken to represent binary 1 and no punching to represent binary 0.

<table>
<thead>
<tr>
<th>Zone 1, No. 8 = CNCL = 001/1000</th>
<th>4 zone bits</th>
<th>4 numeric bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone 4, No. 12 = L = 100/1100</td>
<td>8 4 2 1</td>
<td>1 1</td>
</tr>
<tr>
<td>Zone 5, No. 10 = Z = 101/1010</td>
<td>8 4 2 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

N.B. No parity is indicated above

The actual representation on tape would be as follows:

For full 1900 code see table overleaf
### 8-Track 1900 Paper Tape Code, using 7 Data Bits

(The eighth track is employed only as a parity track, if required)

<table>
<thead>
<tr>
<th>Zone</th>
<th>Numeric</th>
<th>0 (000)</th>
<th>1 (001)</th>
<th>2 (010)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC&lt;sub&gt;0&lt;/sub&gt; (NULL)</td>
<td>TC&lt;sub&gt;0&lt;/sub&gt; Data Link Escape</td>
<td>space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;1&lt;/sub&gt; Start of Heading</td>
<td>DC&lt;sub&gt;2&lt;/sub&gt; Device Control</td>
<td>! (exclamation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;2&lt;/sub&gt; Start of Text</td>
<td>DC&lt;sub&gt;2&lt;/sub&gt;</td>
<td>&quot; (quotes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;3&lt;/sub&gt; End of Text</td>
<td>DC&lt;sub&gt;5&lt;/sub&gt; (STOP)</td>
<td># (number)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;4&lt;/sub&gt; End of Transmission</td>
<td>TC&lt;sub&gt;6&lt;/sub&gt; Negative Acknowledge</td>
<td>£ (pound)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;5&lt;/sub&gt; Enquiry</td>
<td>TC&lt;sub&gt;9&lt;/sub&gt; Synchronous Idle</td>
<td>&amp; (ampersand)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;sub&gt;6&lt;/sub&gt; Acknowledge</td>
<td>TC&lt;sub&gt;9&lt;/sub&gt; End of Transmission Block</td>
<td>' (apostrophe)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEL Bell, alarm</td>
<td>CNCL Cancel</td>
<td>( ) (i.e., parenth.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;9&lt;/sub&gt; Backspace</td>
<td>EM End of Medium</td>
<td>) (r.parenth.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;1&lt;/sub&gt; Horizontal Tabulation</td>
<td>SB Substitute</td>
<td>* (asterisk)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;2&lt;/sub&gt; New Line</td>
<td>ESC Escape</td>
<td>+ (plus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;3&lt;/sub&gt; Line Feed (Start new line)</td>
<td>IS&lt;sub&gt;s&lt;/sub&gt; File Separator</td>
<td>, (comma)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;4&lt;/sub&gt; Form Feed</td>
<td>IS&lt;sub&gt;s&lt;/sub&gt; Group Separator</td>
<td>- (hyphen/minus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE&lt;sub&gt;5&lt;/sub&gt; Carriage Return</td>
<td>IS&lt;sub&gt;2&lt;/sub&gt; Record Separator</td>
<td>. (period)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SO Shift Out</td>
<td>IS&lt;sub&gt;1&lt;/sub&gt; Unit Separator</td>
<td>/ (solidus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI Shift In</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE 1**  
Synchronous Idle: A transmission control character used by a synchronous transmission system in the absence of any other character (Idle condition) to provide a signal from which synchronism may be achieved or retained.

2  
The code as shown above is that used on the 1900 series. It is the Standard I.S.O. 7-bit code, with the following minor exceptions:

- I.S.O. £ has been replaced by #
- I.S.O. $ has been replaced by £
- I.S.O. _ (underline) has been replaced by ←

Diacritical signs (accents) have not been implemented. Codes permitted for national use have been used for $ and _ (underline).

The above exceptions are due to changes in the I.S.O. code since the 1900 series code, originally fully compatible, was finalized.
<table>
<thead>
<tr>
<th>3 (011)</th>
<th>4 (100)</th>
<th>5 (101)</th>
<th>6 (110)</th>
<th>7 (111)</th>
<th>Zone</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>@ (at)</td>
<td>P</td>
<td>_ (underline)</td>
<td>p</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
<td>0010</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
<td>0011</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td>0100</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td>0101</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
<td>0110</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td>0111</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>:) (colon)</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td>1010</td>
<td></td>
</tr>
<tr>
<td>;) (semi colon)</td>
<td>K</td>
<td>[ (l.h. brkt)</td>
<td>k</td>
<td>N1 Reserved for National Characters</td>
<td>1011</td>
<td></td>
</tr>
<tr>
<td>&lt; (less than)</td>
<td>L</td>
<td>$ (dollar)</td>
<td>l</td>
<td>N2</td>
<td>1100</td>
<td></td>
</tr>
<tr>
<td>= (equals)</td>
<td>M</td>
<td>] (r.h. brkt)</td>
<td>m</td>
<td>N3</td>
<td>1101</td>
<td></td>
</tr>
<tr>
<td>&gt; (greater than)</td>
<td>N</td>
<td>↑ o</td>
<td>n</td>
<td>N4</td>
<td>1110</td>
<td></td>
</tr>
<tr>
<td>? (question)</td>
<td>O</td>
<td>← (delete)</td>
<td>o</td>
<td></td>
<td>1111</td>
<td></td>
</tr>
</tbody>
</table>
6-Bit, 3 Shift Internal Machine Form

When using the seven data bit paper-tape code it will be necessary, on input, to condense the seven data bit code into six data bits and also to expand the internal six data bits to seven on output from the store. This is achieved by condensing the three-bit zone identification into a two-bit zone identification as follows:

(a) The middle bit of the zone is dropped and the two remaining bits are compared.

(b) If the bits are equal, a 1 bit is substituted for the right-hand bit and is inserted in position 2^4 of the character; the left-hand bit enters position 2^5 unaltered.

(c) If the bits are unequal, a 0 bit is substituted for the right-hand bit and is inserted in position 2^4 of the character location; the left-hand bit enters position 2^5 unaltered. For example, the zone component for tape character A is condensed as below:

\[
\begin{array}{cccccc}
1 & \neq & 0 & & & \\
1 & 0 & & & & \\
\end{array}
\]

The middle bit is discarded.

Therefore, 0 is generated and inserted in 2^4; the left-hand bit (1) enters 2^5.

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 1 \\
2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
\end{array}
\]

Problems are thus immediately introduced. For example both zones 010 and 000 would have to be condensed to 00. Similarly 101 and 111 would both be represented as 11. The problem, then, clearly is one of identification, and for this purpose three identifying characters known as the shift characters have been introduced. The table opposite shows the internal representation of characters, excluding shift characters.

In order to provide uniquely recognizable bit patterns for the three shift characters when programming for paper tape, the numeric content of certain character codes (namely, the four symbols $\$$, $\#$, $\uparrow$, and the four National characters) is also altered by the paper tape peripherals. Of these characters, those of zone 110 have the 8-numeric bit replaced by a 0 bit; and those of zone 111 (the National characters) have the 4-numeric bit replaced by a 0 bit. See the table on page 11 of this chapter.

For further information see the 1900 Series manual Basic Peripherals.
## The Internal Machine Code, Excluding Shift Characters

<table>
<thead>
<tr>
<th>Internal Code</th>
<th>Zone 0 (00)</th>
<th>Zone 1 (01)</th>
<th>Zone 2 (10)</th>
<th>Zone 3 (11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>space</td>
<td>@ (at)</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>! (exclamation)</td>
<td>A</td>
<td>Q</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>&quot; (quotes)</td>
<td>B</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td># (number)</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>£ (pound)</td>
<td>D</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>% (percentage)</td>
<td>E</td>
<td>U</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>&amp; (ampersand)</td>
<td>F</td>
<td>V</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>' (apostrophe)</td>
<td>G</td>
<td>W</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>( (left parenthesis)</td>
<td>H</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>) (right parenthesis)</td>
<td>I</td>
<td>Y</td>
</tr>
<tr>
<td>10</td>
<td>: (colon)</td>
<td>* (asterisk)</td>
<td>J</td>
<td>Z</td>
</tr>
<tr>
<td>11</td>
<td>; (semi-colon)</td>
<td>+ (plus)</td>
<td>K</td>
<td>[ (l.h. bracket)</td>
</tr>
<tr>
<td>12</td>
<td>&lt; (less than)</td>
<td>, (comma)</td>
<td>L</td>
<td>$ (dollar)</td>
</tr>
<tr>
<td>13</td>
<td>= (equals)</td>
<td>- (hyphen/minus)</td>
<td>M</td>
<td>] (r.h. bracket)</td>
</tr>
<tr>
<td>14</td>
<td>&gt; (greater than)</td>
<td>. (stop)</td>
<td>N</td>
<td>↑</td>
</tr>
<tr>
<td>15</td>
<td>? (question)</td>
<td>/ (solidus)</td>
<td>O</td>
<td>←</td>
</tr>
</tbody>
</table>
## 6-Bit, 3 Shift Internal Machine Form

<table>
<thead>
<tr>
<th>Zone and Shift Numeric</th>
<th>α or β</th>
<th>δ</th>
<th>α or β</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>TC₁ Data Link Escape</td>
<td>space</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>DC₁ Device Control</td>
<td>:</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>DC₂</td>
<td>&quot;</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>DC₃</td>
<td>#</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>DC₄ Stop</td>
<td>£</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>TC₅ Negative Acknowledge</td>
<td>%</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>TC₆ Synchronous Idle</td>
<td>&amp;</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>TC₇ End of Transmission Block</td>
<td>'</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>CNCL Cancel</td>
<td>(</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>EM End of Medium</td>
<td>)</td>
</tr>
<tr>
<td>1010</td>
<td>:</td>
<td>(colon)</td>
<td>*</td>
</tr>
<tr>
<td>1011</td>
<td>;</td>
<td>(semi-colon)</td>
<td>+</td>
</tr>
<tr>
<td>1100</td>
<td>&lt;</td>
<td>(less than)</td>
<td>,</td>
</tr>
<tr>
<td>1101</td>
<td>=</td>
<td>(equals)</td>
<td>-</td>
</tr>
<tr>
<td>1110</td>
<td>&gt;</td>
<td>(greater than)</td>
<td>.</td>
</tr>
<tr>
<td>1111</td>
<td>?</td>
<td>(question)</td>
<td>/</td>
</tr>
</tbody>
</table>

**Explanation**

- **TC₁ Data Link Escape**
- **DC₁ Device Control**
- **DC₂**
- **DC₃**
- **DC₄ Stop**
- **TC₅ Negative Acknowledge**
- **TC₆ Synchronous Idle**
- **TC₇ End of Transmission Block**
- **CNCL Cancel**
- **EM End of Medium**
- **(colon)**
- **(semi-colon)**
- **< (less than)**
- **= (equals)**
- **> (greater than)**
- **? (question)**

**Note:**

- **space**
- **: (exclamation)**
- **" (quotes)**
- **# (number)**
- **£ (pound)**
- **% (percent)**
- **& (ampersand)**
- **' (apostrophe)**
- **( (left parenthesis)**
- **) (right parenthesis)**
- *** (asterisk)**
- **+ (plus)**
- **, (comma)**
- **- (hyphen/minus)**
- **. (stop)**
- **/ (solidus)**
<table>
<thead>
<tr>
<th>01 continued</th>
<th>10</th>
<th>11</th>
<th>Zone and Shift Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>δ</td>
<td>α</td>
<td>β</td>
<td>δ</td>
</tr>
<tr>
<td>TC₀ Null</td>
<td>φ</td>
<td>_</td>
<td>P</td>
</tr>
<tr>
<td>TC₁ Start of heading</td>
<td>A</td>
<td>a</td>
<td>Q</td>
</tr>
<tr>
<td>TC₂ Start of text</td>
<td>B</td>
<td>b</td>
<td>R</td>
</tr>
<tr>
<td>TC₃ End of text</td>
<td>C</td>
<td>c</td>
<td>S</td>
</tr>
<tr>
<td>TC₄ End of Transmission</td>
<td>D</td>
<td>d</td>
<td>T</td>
</tr>
<tr>
<td>TC₅ Enquiry</td>
<td>E</td>
<td>e</td>
<td>U</td>
</tr>
<tr>
<td>TC₆ Acknowledge</td>
<td>F</td>
<td>f</td>
<td>V</td>
</tr>
<tr>
<td>BEL Bell, Alarm</td>
<td>G</td>
<td>g</td>
<td>W</td>
</tr>
<tr>
<td>FE₀ Back Space</td>
<td>H</td>
<td>h</td>
<td>X</td>
</tr>
<tr>
<td>FE₁ Horizontal Tabulation</td>
<td>I</td>
<td>i</td>
<td>Y</td>
</tr>
<tr>
<td>FE₂ Newline</td>
<td>J</td>
<td>j</td>
<td>Z</td>
</tr>
<tr>
<td>FE₃ Line Feed (Start New Line)</td>
<td>K</td>
<td>k</td>
<td>] (l.h. brkt)</td>
</tr>
<tr>
<td>FE₄ Form Feed</td>
<td>L</td>
<td>l</td>
<td>α</td>
</tr>
<tr>
<td>FE₅ Carriage Return</td>
<td>M</td>
<td>m</td>
<td>β</td>
</tr>
<tr>
<td>SO Shift Out</td>
<td>N</td>
<td>n</td>
<td>δ</td>
</tr>
<tr>
<td>SI Shift In</td>
<td>O</td>
<td>o</td>
<td>_</td>
</tr>
</tbody>
</table>

Notes

1. If it is required to use the heading facility for paper tape output it should be noted that when the symbols $ \uparrow \leftarrow $ are read by the PLAN compiler, preceded by nH, they are converted to characters under internal code.

   δ $ becomes 11 1100
   δ ] becomes 11 1101
   δ ↑ becomes 11 1110
   δ ← becomes 11 1111

2. N₁, N₂, N₃, N₄ represent national characters.
The three shift characters will be represented as follows; alpha (α) by octal 74 (111100); beta (β) by octal 75 (111101) and delta (δ) by octal 76 (111110), these three notations corresponding to positions 12, 13 and 14 of Zone 7 of the 1900 paper tape punching code. When input is read from paper tape the shift characters are automatically inserted into the store. The machine code, when preceded by the appropriate shift character, will be automatically converted on output to the 1900 paper tape code. If, however, input is from any other media (e.g. magnetic tape and punched cards) and output onto paper tape is required, the shift character must be inserted by program.

In the table it will be noticed that in Zones 00 and 01 either the α or the β character serves to identify the characters in the left hand columns. In Zones 10 and 11, however, the α shift character is used to denote upper case and the β shift character to denote lower case. This provision has been made because the characters in the left hand columns of Zones 00 and 01 will be required whether output is in upper or lower case. On input a character in alpha shift will only produce an alpha shift character to precede it if the reader is not already in alpha shift. (The program may set alpha shift before reading commences.) Similarly if a character in beta shift is read, and the reader is not already in beta shift, then a beta shift character is inserted before the character in store. Once in the new shift subsequent characters do not cause shift characters to be inserted unless they change the shift. The delta shift character refers only to the next single character and, immediately after that character, the reader reverts automatically to the shift it was in before the delta shift was encountered. Punched output to paper tape is the reverse of the above description from the reader.

**Comparison with 7-Data Bit Paper Tape Code**

<table>
<thead>
<tr>
<th>Shift Character and Internal Zone</th>
<th>Paper Tape Code Zone</th>
<th>Shift Character and Internal Zone</th>
<th>Paper Tape Code Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>α or β 00</td>
<td>3</td>
<td>β 10</td>
<td>6</td>
</tr>
<tr>
<td>δ 00</td>
<td>1</td>
<td>δ 10</td>
<td>-</td>
</tr>
<tr>
<td>α or β 01</td>
<td>2</td>
<td>α 11</td>
<td>5</td>
</tr>
<tr>
<td>δ 01</td>
<td>0</td>
<td>β 11</td>
<td>7</td>
</tr>
<tr>
<td>α 10</td>
<td>4</td>
<td>δ 11</td>
<td>-</td>
</tr>
</tbody>
</table>
**Printer Code**

The chart below shows the response of the line printer to the six-bit internal machine code.

<table>
<thead>
<tr>
<th>Least Significant 3 bits</th>
<th>Most Significant 3 bits of 6-bit Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0  1  2  3  4  5  6  7</td>
</tr>
<tr>
<td>0</td>
<td>8 SPACE ( @ H P X</td>
</tr>
<tr>
<td>1</td>
<td>9 ! ) A I Q Y</td>
</tr>
<tr>
<td>2</td>
<td>: &quot; * B J R Z</td>
</tr>
<tr>
<td>3</td>
<td>; # + C K S ]</td>
</tr>
<tr>
<td>4</td>
<td>&lt; , D L T $</td>
</tr>
<tr>
<td>5</td>
<td>= - E M U ¥</td>
</tr>
<tr>
<td>6</td>
<td>&gt; &amp; . F N V ↑</td>
</tr>
<tr>
<td>7</td>
<td>? / G O W ←</td>
</tr>
</tbody>
</table>
# Appendix 2 Tables of Powers of 2

## Table of Powers of 2 (Decimal)

<table>
<thead>
<tr>
<th>(2^n)</th>
<th>(n)</th>
<th>(2^n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0.125</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>0.062 5</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>0.031 25</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>0.015 625</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
<td>0.007 812 5</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>0.003 906 25</td>
</tr>
<tr>
<td>512</td>
<td>9</td>
<td>0.001 953 125</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
<td>0.000 976 562 5</td>
</tr>
<tr>
<td>2048</td>
<td>11</td>
<td>0.000 494 501 25</td>
</tr>
<tr>
<td>4096</td>
<td>12</td>
<td>0.000 244 160 625</td>
</tr>
<tr>
<td>8192</td>
<td>13</td>
<td>0.000 122 070 312 5</td>
</tr>
<tr>
<td>16384</td>
<td>14</td>
<td>0.000 061 035 156 25</td>
</tr>
<tr>
<td>32768</td>
<td>15</td>
<td>0.000 030 517 678 125</td>
</tr>
<tr>
<td>65536</td>
<td>16</td>
<td>0.000 015 258 759 062 5</td>
</tr>
<tr>
<td>131072</td>
<td>17</td>
<td>0.000 007 629 394 531 25</td>
</tr>
<tr>
<td>262144</td>
<td>18</td>
<td>0.000 003 814 897 265 625</td>
</tr>
<tr>
<td>524288</td>
<td>19</td>
<td>0.000 001 907 346 632 812 5</td>
</tr>
<tr>
<td>1048576</td>
<td>20</td>
<td>0.000 000 953 674 318 406 25</td>
</tr>
<tr>
<td>2097152</td>
<td>21</td>
<td>0.000 000 476 837 158 203 125</td>
</tr>
<tr>
<td>4194304</td>
<td>22</td>
<td>0.000 000 238 418 579 101 562 5</td>
</tr>
<tr>
<td>8388608</td>
<td>23</td>
<td>0.000 000 119 209 299 590 781 25</td>
</tr>
<tr>
<td>16777216</td>
<td>24</td>
<td>0.000 000 059 804 644 775 390 625</td>
</tr>
<tr>
<td>33554432</td>
<td>25</td>
<td>0.000 000 029 802 322 387 965 312 5</td>
</tr>
<tr>
<td>67108864</td>
<td>26</td>
<td>0.000 000 014 901 618 193 847 656 25</td>
</tr>
<tr>
<td>134217728</td>
<td>27</td>
<td>0.000 000 007 456 390 596 923 828 125</td>
</tr>
<tr>
<td>268435456</td>
<td>28</td>
<td>0.000 000 003 725 290 298 461 914 063</td>
</tr>
<tr>
<td>536870912</td>
<td>29</td>
<td>0.000 000 001 862 645 149 230 857 031</td>
</tr>
<tr>
<td>1073741824</td>
<td>30</td>
<td>0.000 000 000 931 352 774 615 476 316</td>
</tr>
<tr>
<td>2147483648</td>
<td>31</td>
<td>0.000 000 000 465 661 287 307 739 258</td>
</tr>
<tr>
<td>4294967296</td>
<td>32</td>
<td>0.000 000 000 232 830 643 652 809 629</td>
</tr>
<tr>
<td>8589934592</td>
<td>33</td>
<td>0.000 000 000 116 415 321 896 934 814</td>
</tr>
<tr>
<td>17179869184</td>
<td>34</td>
<td>0.000 000 000 058 207 660 913 467 407</td>
</tr>
<tr>
<td>34359738368</td>
<td>35</td>
<td>0.000 000 000 029 103 830 456 733 704</td>
</tr>
<tr>
<td>68719476736</td>
<td>36</td>
<td>0.000 000 000 014 551 915 228 386 852</td>
</tr>
<tr>
<td>137438953472</td>
<td>37</td>
<td>0.000 000 000 007 275 957 614 183 426</td>
</tr>
<tr>
<td>274877906944</td>
<td>38</td>
<td>0.000 000 000 003 637 978 097 091 713</td>
</tr>
<tr>
<td>549755813888</td>
<td>39</td>
<td>0.000 000 000 001 918 969 403 545 856</td>
</tr>
<tr>
<td>1099511627776</td>
<td>40</td>
<td>0.000 000 000 000 909 494 701 772 925</td>
</tr>
<tr>
<td>2199023555552</td>
<td>41</td>
<td>0.000 000 000 000 454 747 350 846 464</td>
</tr>
<tr>
<td>4398046111056</td>
<td>42</td>
<td>0.000 000 000 000 227 373 675 643 232</td>
</tr>
<tr>
<td>8796092222112</td>
<td>43</td>
<td>0.000 000 000 000 113 688 837 721 616</td>
</tr>
<tr>
<td>17592199444224</td>
<td>44</td>
<td>0.000 000 000 000 056 843 418 960 809</td>
</tr>
<tr>
<td>35184398888448</td>
<td>45</td>
<td>0.000 000 000 000 028 421 709 436 404</td>
</tr>
<tr>
<td>70368797777896</td>
<td>46</td>
<td>0.000 000 000 000 014 210 854 715 202</td>
</tr>
<tr>
<td>140737488855792</td>
<td>47</td>
<td>0.000 000 000 000 007 105 427 357 801</td>
</tr>
</tbody>
</table>

---

Appendix 2
### Decimal → Octal Conversion Table

<table>
<thead>
<tr>
<th>Digit Value</th>
<th>8th</th>
<th>7th</th>
<th>6th</th>
<th>5th</th>
<th>4th</th>
<th>3rd</th>
<th>2nd</th>
<th>1st</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46113200</td>
<td>3641100</td>
<td>303240</td>
<td>23420</td>
<td>1750</td>
<td>144</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>114226400</td>
<td>7502200</td>
<td>606500</td>
<td>47040</td>
<td>3720</td>
<td>310</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>13343300</td>
<td>1111740</td>
<td>72460</td>
<td>5670</td>
<td>454</td>
<td>36</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>17204400</td>
<td>1415200</td>
<td>116100</td>
<td>7640</td>
<td>620</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>23045500</td>
<td>17204400</td>
<td>1415200</td>
<td>116100</td>
<td>764</td>
<td>62</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>26706600</td>
<td>2223700</td>
<td>165140</td>
<td>13560</td>
<td>1130</td>
<td>74</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>32547700</td>
<td>2527140</td>
<td>210560</td>
<td>15530</td>
<td>1274</td>
<td>106</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>36411000</td>
<td>3032400</td>
<td>234200</td>
<td>17500</td>
<td>1440</td>
<td>120</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>42252100</td>
<td>3335640</td>
<td>257620</td>
<td>21450</td>
<td>1604</td>
<td>132</td>
<td>11</td>
</tr>
</tbody>
</table>

**NOTES**

1. The first digit is the least significant.
2. When performing the addition the base is 8 and not 10 as in normal decimal addition.
   
   i.e. 6 \( \neq \) 6 \[ \text{e.g. } 16547328 = 10 \]
   
   \[ \begin{array}{c}
   +4 \\
   \hline
   10 \\
   +4 \\
   \hline
   14 \\
   \end{array} \]

### Octal → Decimal Conversion Table

<table>
<thead>
<tr>
<th>Digit Value</th>
<th>8th</th>
<th>7th</th>
<th>6th</th>
<th>5th</th>
<th>4th</th>
<th>3rd</th>
<th>2nd</th>
<th>1st</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2097152</td>
<td>262144</td>
<td>32768</td>
<td>4096</td>
<td>512</td>
<td>64</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4194304</td>
<td>524288</td>
<td>65536</td>
<td>8192</td>
<td>1024</td>
<td>128</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6291456</td>
<td>786432</td>
<td>98304</td>
<td>12288</td>
<td>1536</td>
<td>192</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8388608</td>
<td>1048576</td>
<td>131072</td>
<td>16384</td>
<td>2048</td>
<td>256</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>10485760</td>
<td>1310720</td>
<td>163840</td>
<td>20480</td>
<td>2560</td>
<td>320</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>12582912</td>
<td>1572864</td>
<td>196608</td>
<td>24576</td>
<td>3072</td>
<td>384</td>
<td>48</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>14680064</td>
<td>1835008</td>
<td>229376</td>
<td>28672</td>
<td>3584</td>
<td>448</td>
<td>56</td>
<td>7</td>
</tr>
</tbody>
</table>

**NOTE**

The first digit is the least significant,

\[ \text{e.g. } \# 56347254 = 4 \]

\[ \begin{array}{c}
40 \\
128 \\
3584 \\
16384 \\
98304 \\
1572864 \\
10485760 \\
\hline
12177068 \\
\end{array} \]
Appendix 4  Format of Control Area for Use with 'PERI' Instructions
### Character Peripherals - Transfer Operations

<table>
<thead>
<tr>
<th>Control Area</th>
<th>Paper Tape Reader</th>
<th>Paper Tape Punch</th>
<th>Line Printer†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 0 to 8</td>
<td>2° on - read N characters or to stop character</td>
<td>4° on - start in previous shift</td>
<td>1 Paper advance and/or print one line</td>
</tr>
<tr>
<td>Bits 9 to 23</td>
<td>off - read N characters</td>
<td>off - start in α shift</td>
<td>2 Automatic write</td>
</tr>
<tr>
<td></td>
<td>4° on - start in previous shift</td>
<td>8° on - graphic set modes</td>
<td>(Available only on 1933/1 printer).</td>
</tr>
<tr>
<td></td>
<td>off - start in α shift</td>
<td>off - shift modes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8° on - graphic set modes</td>
<td>16° on - all characters mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>off - shift modes</td>
<td>off - normal mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16° on - all characters mode</td>
<td>ignoring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>off - normal mode</td>
<td>blank and shift</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ignoring</td>
<td>erase</td>
<td></td>
</tr>
<tr>
<td></td>
<td>blank and</td>
<td>mode</td>
<td></td>
</tr>
<tr>
<td>Word 1 Reply</td>
<td>0 = Transfer completed</td>
<td>0 = Transfer completed</td>
<td>0 = Transfer completed</td>
</tr>
<tr>
<td>Bit 0</td>
<td>(i.e. - Reply word set negative.) = Transfer in progress</td>
<td>(i.e. - Reply word set negative.) = Transfer in progress</td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>Undefined</td>
<td>Undefined</td>
<td>1 Overflow</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Undefined</td>
<td>Undefined</td>
<td>1 Paper low</td>
</tr>
<tr>
<td>Bit 3</td>
<td>1 = Transfer error</td>
<td>1 = Transfer error</td>
<td>1 = Transfer error</td>
</tr>
<tr>
<td>Bits 4 to 16</td>
<td>Undefined</td>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>Bits 17 to 23</td>
<td>Remainder character count</td>
<td>Remainder character count if B3 = 1.</td>
<td>Remainder character count if B3 = 0.</td>
</tr>
</tbody>
</table>

### Word 2 Count

| Character count. Maximum 128 | Character count. Maximum 128 | Character count. For maximum count and print control character see next page |

### Word 3 Core Address of Start of Data

| Character address may be specified | Character address may be specified | Character address. For mode 0 this must be character address 3. |

---

† Line Printer Control Characters

**Mode 0**

The first character of any string of characters transferred to the line printer in mode 0 is taken as the print control character. It should have one of the following values, regarded as an octal digit pair:

<table>
<thead>
<tr>
<th>No Print</th>
<th>Print</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>40</td>
</tr>
<tr>
<td>01</td>
<td>41</td>
</tr>
<tr>
<td>02</td>
<td>42</td>
</tr>
<tr>
<td>11</td>
<td>51</td>
</tr>
<tr>
<td>12 to 17</td>
<td>52 to 57</td>
</tr>
<tr>
<td>Card Reader</td>
<td>Card Punch</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0 Normal decode mode</td>
<td>Only available mode</td>
</tr>
<tr>
<td>2 Column image mode</td>
<td></td>
</tr>
<tr>
<td>(Two characters per column. Available only on the 2103 reader and readers fitted with the Binary Image facility.)</td>
<td></td>
</tr>
<tr>
<td>32* Inhibit code translation (Interface code is transferred instead of translated code. Available only with the 2102 reader, basic mode 0, and the 2103 reader, basic modes 0 and 2.)</td>
<td></td>
</tr>
<tr>
<td>0 = Transfer completed</td>
<td>0 = Transfer completed</td>
</tr>
<tr>
<td>1 (i.e. - Reply word set negative.) = Transfer in progress</td>
<td>1 (i.e. - Reply word set negative.) = Transfer in progress</td>
</tr>
<tr>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>1 = Transfer error</td>
<td>1 = Transfer error</td>
</tr>
<tr>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>Mode 0 80 characters</td>
<td>Maximum 80</td>
</tr>
<tr>
<td>Mode 2 160 characters</td>
<td>80 characters always punched for values 1 to 80</td>
</tr>
<tr>
<td>Word address only</td>
<td>Word address only</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The control character must be character 3 of a word. The print transfer commences from this character and 1 should be added for the control character to the count of the number of characters to be printed. The maximum value of the count is 161. On buffered printers the number of characters actually printed will be the number given in the count minus 1, subject to the maximum number of print positions available on the printer. On unbuffered printers the number of characters printed will be determined by the number of print positions on the printer (for all count values exceeding 1).

Mode 2

In Mode 2, automatic write, the data character - (octal 77) is not printed, but indicates that the following one or more characters are control characters. Using automatic write one print order may be used to print several consecutive lines. The character count is the total of printable and control characters to be sent to the printer. There is no maximum value other that imposed by the program's limit in store. Full details of the automatic write facility are given in the Basic Peripherals manual.
### Magnetic and Cassette Tape - Transfer Operations

<table>
<thead>
<tr>
<th>Control Area</th>
<th>Magnetic Tape</th>
<th>Cassette Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Word 0</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 0 to 8: Device Type</td>
<td>0 Read</td>
<td>0 Read</td>
</tr>
<tr>
<td>Bits 9 to 23: Mode</td>
<td>1 Write</td>
<td>1 Write</td>
</tr>
<tr>
<td></td>
<td>2 Read reverse</td>
<td>2 Erase a section of tape</td>
</tr>
<tr>
<td></td>
<td>3 Backspace</td>
<td>7 Align</td>
</tr>
<tr>
<td></td>
<td>4 Skip forward past tape mark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 Write tape mark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 Skip back past tape mark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 Rewind</td>
<td></td>
</tr>
<tr>
<td><strong>Word 1 Reply</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>0 = Operation completed</td>
<td>0 = Operation completed</td>
</tr>
<tr>
<td></td>
<td>1 (i.e. - Reply word set negative,) = Operation in progress</td>
<td>1 (i.e. - Reply word set negative,) = Operation in progress</td>
</tr>
<tr>
<td>Bit 1</td>
<td>1 = Long block (modes 0,2)</td>
<td>1 = Long block (mode 0)</td>
</tr>
<tr>
<td>Bit 2</td>
<td>1 = Corrected in-flight error (9-track tape, modes 0 and 2), or Undefined (7-track tape, and other modes of 9-track tape)</td>
<td>Undefined</td>
</tr>
<tr>
<td>Bit 3</td>
<td>1 = Transfer error</td>
<td>1 = Transfer error</td>
</tr>
<tr>
<td>Bit 4</td>
<td>1 = End of tape (modes 1,5) or Tape mark (modes 0,2), or Undefined (modes 3,4,6,7)</td>
<td>1 = End of tape (modes 0,1,2)</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Undefined</td>
<td>Undefined</td>
</tr>
<tr>
<td>Bits 6 to 23</td>
<td>Address (18 bits) of next word to be read or written if transfer had not terminated (modes 0,1,2 only) or undefined (modes 0 and 2 if bit 4 is set, and modes 3 to 7) except as indicated in note 6.</td>
<td>Address (18 bits) of next word to be read or written if transfer had not terminated (modes 0,1)</td>
</tr>
<tr>
<td><strong>Word 2 Count</strong></td>
<td>Word count</td>
<td>Word count (excluding block sequence).</td>
</tr>
<tr>
<td></td>
<td>Maximum 32,767</td>
<td>Maximum 512</td>
</tr>
<tr>
<td></td>
<td>(Modes 0,1,2 only)</td>
<td>(Modes 0,1)</td>
</tr>
<tr>
<td><strong>Word 3</strong></td>
<td>Word address only</td>
<td>Word address only</td>
</tr>
<tr>
<td>Core Address of Start of Data</td>
<td>(Modes 0,1,2 only)</td>
<td>(second word of data area)</td>
</tr>
<tr>
<td></td>
<td>(Modes 0,1,2 only)</td>
<td>(Modes 0,1 only)</td>
</tr>
<tr>
<td><strong>Word 4</strong></td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Device Address of Start of Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 For mode 320 on direct access devices the third word is used as additional reply (initially zero). Fourth word specifies number of blocks and fifth word contains, optionally, magazine/cartridge serial number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 For close file modes only words 1 and 2 are required in the control area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 The additive mode 14336 (#34000) may be used with magnetic tape open tape modes if Own Monitor is required. See the Magnetic Tape manual.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 For further details of reply information with direct access devices and the magnetic drum, see the Direct Access manual.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 For direct access devices, mode 320: Word 2 Additional reply word. Word 3 Number of blocks by which the file is to be expanded (positive number) or contracted (negative number). Word 4 Serial number (optional) or zero. The file must already have been opened as a write file. For further detail see the Direct Access manual.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 For modes 64, 128, on the magnetic drum, word 5 contains the file size on completion of open.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Magnetic and Cassette Tape - Open and Close Operations

<table>
<thead>
<tr>
<th>Control Area</th>
<th>Magnetic Tape and Cassette Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 0 Type</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>#100 Open labelled tape for reading No check on write permit ring</td>
</tr>
<tr>
<td></td>
<td>#200 Open labelled tape for reading Check write permit ring absent</td>
</tr>
<tr>
<td></td>
<td>#300 Open labelled tape for reading and writing</td>
</tr>
<tr>
<td></td>
<td>#400 Open output tape and label for retention</td>
</tr>
<tr>
<td></td>
<td>#500 Relabel opened tape</td>
</tr>
<tr>
<td></td>
<td>#600 Open output tape and label it 'Scratch Tape' with zero retention period</td>
</tr>
<tr>
<td></td>
<td>#1000 Close file (Rewind: de-allocate deck but do not disengage)</td>
</tr>
<tr>
<td></td>
<td>#1007 Unload file (Rewind and disengage deck)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 1 Reply</th>
<th>Open Modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>0 = Tape successfully opened</td>
</tr>
<tr>
<td></td>
<td>1 = Additive mode 14336 has been used, and the tape has not been successfully opened.</td>
</tr>
<tr>
<td>Bits 1 to 23</td>
<td>Tape serial number if the tape has been successfully opened; if the tape has not been successfully opened, undefined</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Close Modes:</th>
<th>Bits 0 to 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Modes:</td>
<td>Undefined</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 5</th>
<th>Reel serial number (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 6</td>
<td>File generation number (binary)</td>
</tr>
<tr>
<td>Word 7</td>
<td>Retention period (days, binary) (with security bit)</td>
</tr>
<tr>
<td>Word 8</td>
<td>Date written (days from 1.1.1900, binary)</td>
</tr>
<tr>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>1 If 256 is added to the type, then the X address of the open/close order is taken to refer to an accumulator containing the file number.</td>
<td></td>
</tr>
<tr>
<td>2 For close file modes only words 1 and 2 are required in the Control area.</td>
<td></td>
</tr>
<tr>
<td>3 The additive mode 14336 (#34000) may be used with magnetic tape open tape modes if Own Monitor is required. See the Magnetic Tape manual.</td>
<td></td>
</tr>
</tbody>
</table>
### Direct Access Devices—Transfer Operations

<table>
<thead>
<tr>
<th>Control Area</th>
<th>Basic Direct Access Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 0</td>
<td>E.D.S.6, M.C.F.7, F.D.S.13</td>
</tr>
<tr>
<td>Bits 0 to 8:</td>
<td>0: Read n words (maximum 1 bucket)</td>
</tr>
<tr>
<td>Bits 9 to 23:</td>
<td>1: Write n words (maximum 1 bucket)</td>
</tr>
<tr>
<td></td>
<td>2: Read n words (maximum 1 seek area)</td>
</tr>
<tr>
<td></td>
<td>3: Write n words (maximum 1 seek area)</td>
</tr>
<tr>
<td>#40</td>
<td>(Additive mode) Suppress check read (Null on M.C.F.)</td>
</tr>
</tbody>
</table>

| Word 1 Reply | 0 = Transfer completed |
| Bit 0        | 1 = End of seek area reached (modes 2 and 3) |
| Bit 1        | 1 = Auxiliary control area full |
| Bit 2        | 1 = Unclearable parity error |
| Bit 3        | 1 = Card has been accessed n times and should be replaced (M.C.F. only) |
| Bit 4        | 1 = Bucket number out of range or zero |
| Bits 6 to 23 | Address (18 bits) of next word to be read or written if transfer had not terminated, if Bit 1 = 1 or Bit 4 = 1. Otherwise undefined. |

| Word 2 Count | Word Count. Maximum 1 bucket (modes 0, 1) or 1 seek area (modes 2, 3) |

<p>| Word 3 Core Address of Start of Data | Word address only |
| Word 4 Device Address of Start of Data | Logical bucket number (in least significant 23 bits) |</p>
<table>
<thead>
<tr>
<th>Unified Direct Access Standard</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.D.S., F.D.S.6</td>
<td></td>
</tr>
<tr>
<td>0    Read n words (maximum 1 bucket)</td>
<td>1</td>
</tr>
<tr>
<td>1    Write n words (maximum 1 bucket)</td>
<td>2</td>
</tr>
<tr>
<td>2    Read n words (maximum 1 seek area)</td>
<td></td>
</tr>
<tr>
<td>3    Write n words (maximum 1 seek area)</td>
<td></td>
</tr>
<tr>
<td>7    More read/write head to given bucket number</td>
<td></td>
</tr>
<tr>
<td>40   (Additive mode) Suppress check read after write</td>
<td></td>
</tr>
</tbody>
</table>

| 0 = Transfer completed       |       |
| 1 (that is, Reply word set negative) | 2     | With UDAS if Bit 3=1 only Bit 2 is defined |
| = Transfer in progress       |       |
| 1 = End of seek area reached (modes 2 and 3) | 3     | With UDAS if Bit 5=1, Bits 0 to 4 are zero |
| 1 = Flaw area full           |       |
| 1 = Unclearable parity error |       |
| Undefined                    |       |
| 1 = Bucket number out of range or zero Address (18 bits) if next word to be read or written if transfer had not terminated, if Bit 1 = 1. Otherwise undefined. |       |

| Word Count. Maximum 1 bucket (modes 0, 1) or 1 seek area (modes 2, 3) | 4     | For mode 7 this can be any value which would be acceptable for mode 0 |
| Word address only           | 5     | For mode 7 this can be any value which would be acceptable for mode 0 |
| Logical bucket number (in least significant 23 bits) |       |       |
### Direct Access Devices—Open and Close Operations

<table>
<thead>
<tr>
<th>Control Area</th>
<th>Basic Direct Access Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 0</td>
<td>E.D.S.6, M.C.F.7, F.D.S.13</td>
</tr>
<tr>
<td>Bits 0 to 8:</td>
<td>Type</td>
</tr>
<tr>
<td>Bits 9 to 23:</td>
<td>Mode</td>
</tr>
<tr>
<td>#100</td>
<td>Open file in overlay mode</td>
</tr>
<tr>
<td>#200</td>
<td>Open file in input mode</td>
</tr>
<tr>
<td>#300</td>
<td>Open file in output mode</td>
</tr>
<tr>
<td>#400</td>
<td>Open file at end-of-file</td>
</tr>
<tr>
<td>#600</td>
<td>Open scratch file</td>
</tr>
<tr>
<td>#1000</td>
<td>Close file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 1 Reply</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>File successfully opened</td>
</tr>
<tr>
<td>1</td>
<td>File not successfully opened</td>
</tr>
<tr>
<td>Bits 1 to 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bits 6 to 23</td>
<td>Bit 0 = 0: Serial number of unit containing control information for this file Bit 0 = 1: A number identifying the cause of the failure. See Direct Access Manual.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Modes #100,</td>
<td></td>
</tr>
<tr>
<td>#200, #300,</td>
<td>Name of file to be opened</td>
</tr>
<tr>
<td>#400:</td>
<td></td>
</tr>
<tr>
<td>Mode #600:</td>
<td>Bucket size required in blocks</td>
</tr>
<tr>
<td>Mode #1000:</td>
<td>Adjustment to end-of-file tag if file opened for writing. Otherwise zero.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Modes #100,</td>
<td></td>
</tr>
<tr>
<td>#200, #300,</td>
<td>as Word 2</td>
</tr>
<tr>
<td>#400:</td>
<td></td>
</tr>
<tr>
<td>Mode #600:</td>
<td>Number of blocks required</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Modes #100,</td>
<td></td>
</tr>
<tr>
<td>#200, #300,</td>
<td>as Word 2</td>
</tr>
<tr>
<td>#400:</td>
<td></td>
</tr>
<tr>
<td>Mode #600:</td>
<td>Serial number of unit to be used. Otherwise zero.</td>
</tr>
</tbody>
</table>

<p>| Word 5       | Ignored                             |</p>
<table>
<thead>
<tr>
<th>Unified Direct Access Standards</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.D.S., F.D.S.6</td>
<td></td>
</tr>
<tr>
<td>#100 Open file in overlay mode</td>
<td>1 If 256 is added to the type then the X address if the order is taken to refer to an accumulator containing the file number.</td>
</tr>
<tr>
<td>#200 Open file in input mode</td>
<td>2 The additives modes #2000, #6000 may be used with UDAS open modes #100, #200, #300, #400, #1200 to force the use of a storage unit whose serial number is held in Word 1. See Direct Access Manual.</td>
</tr>
<tr>
<td>#300 Open file in output mode</td>
<td></td>
</tr>
<tr>
<td>#400 Open file at end-of-file</td>
<td>3 Additive mode #400 may be used with UDAS modes #600, #1300 to prevent suspension if the required storage unit is off-line. See Direct Access Manual.</td>
</tr>
<tr>
<td>#600 Open scratch file</td>
<td></td>
</tr>
<tr>
<td>#1000 Close file</td>
<td>4 For open modes #100, #200, #300, #400 information about the file is set into words 5 to 8 by Executive after execution.</td>
</tr>
<tr>
<td>#1200 Create and open file</td>
<td></td>
</tr>
<tr>
<td>#1300 Close and delete file</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = File successfully opened</td>
<td>5 The reply is indeterminate for close mode #1000, #1300</td>
</tr>
<tr>
<td>1 = File not successfully opened</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Bit 0 = 0</td>
<td>6 The reply in bits 1-23 for BDAS open mode #600 is</td>
</tr>
<tr>
<td>Serial number of unit containing control information for this file</td>
<td></td>
</tr>
<tr>
<td>Bit 0 = 1</td>
<td>Bit 0 = 0 indeterminate</td>
</tr>
</tbody>
</table>

| Modes #100, #200, #300, #400: Name of file to be opened | 8 Words 2 to 16 of control area for UDAS mode #1200 are set to contain the file specification. See Direct Access Manual. |
| Mode #600: Bucket size required in blocks |       |
| Mode #1000: Adjustment to end-of-file tag if file opened for writing. Otherwise zero |       |
|               | 9 For close file mode #1300 only Words 0 and 1 are required in the control area. |

| Modes #100, #200, #300, #400: as Word 2 | 10 For close file mode #100 only Words 0 to 2 are required in the control area. |
| Mode #600: Number of blocks required |       |

<p>| Modes #100, #200, #300, #400: as Word 2 |       |
| Mode #600: Serial number of unit to be used: otherwise zero |       |
| Ignored | 11 For open mode #600 only Words 0 to 4 are required in the control area. |</p>
<table>
<thead>
<tr>
<th>Control area</th>
<th>Basic Direct Access Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 6</td>
<td>Modes #100, #200, #300, #400:</td>
</tr>
<tr>
<td></td>
<td>File generation number</td>
</tr>
<tr>
<td>Word 7</td>
<td>Modes #100, #200, #400:</td>
</tr>
<tr>
<td></td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Mode #300:</td>
</tr>
<tr>
<td></td>
<td>Data retention period</td>
</tr>
<tr>
<td>Word 8</td>
<td>Modes #100, #200, #400:</td>
</tr>
<tr>
<td></td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Mode #300:</td>
</tr>
<tr>
<td></td>
<td>New generation number</td>
</tr>
<tr>
<td>Unified Direct Access Standards</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Modes #100, #200, #300, #400:</td>
<td>12</td>
</tr>
<tr>
<td>File generation number</td>
<td>For further information on the action taken on the value of Word 6 for UDAS modes #100, #200, #300, #400 see Direct Access Manual</td>
</tr>
<tr>
<td>Modes #100, #200, #400:</td>
<td></td>
</tr>
<tr>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>Mode #300:</td>
<td></td>
</tr>
<tr>
<td>Data retention period</td>
<td></td>
</tr>
<tr>
<td>Modes #100, #200, #400:</td>
<td></td>
</tr>
<tr>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>Mode #300:</td>
<td></td>
</tr>
<tr>
<td>Bit 0 = 0</td>
<td>New generation number</td>
</tr>
<tr>
<td>Bit 0 = 1</td>
<td>Generation number will be left unchanged</td>
</tr>
</tbody>
</table>
### Direct Access Devices—File manipulation operations

<table>
<thead>
<tr>
<th>Control area</th>
<th>Basic Direct Access Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Word 0</strong></td>
<td></td>
</tr>
<tr>
<td>Bits 0 to 8:</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>E.D.S.6, M.C.F.7, F.D.S.13</td>
</tr>
<tr>
<td>Bits 9 to 23:</td>
<td>Mode</td>
</tr>
<tr>
<td></td>
<td>#500  Extend or contract file</td>
</tr>
<tr>
<td></td>
<td>#700  Rename file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Word 1 Reply</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>0 = Action successful</td>
</tr>
<tr>
<td></td>
<td>1 = Action not successful</td>
</tr>
<tr>
<td>Bits 1 to 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bits 6 to 23</td>
<td>Bit 0 = 0</td>
</tr>
<tr>
<td></td>
<td>Mode #500: Last logical bucket number in file before extension or after contraction.</td>
</tr>
<tr>
<td></td>
<td>Mode #700: Indeterminate</td>
</tr>
<tr>
<td></td>
<td>Bit 0 = 1</td>
</tr>
<tr>
<td></td>
<td>A number identifying the cause of the failure.</td>
</tr>
</tbody>
</table>

| **Word 2** | Mode #500: Zero |
|           | Mode #700: New file name |

| **Word 3** | Mode #500: Number of blocks by which file is to be extended or contracted. |
|           | Mode #700: As Word 2 |

<p>| <strong>Word 4</strong> | Mode #500: Serial number of storage unit onto which file is to be extended: otherwise zero. |
|           | Mode #700: As word 2 |</p>
<table>
<thead>
<tr>
<th>Unified Direct Access Standards</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.D.S., F.D.S.6</td>
<td></td>
</tr>
<tr>
<td>#500 Extend or contract file</td>
<td>1</td>
</tr>
<tr>
<td>#700 Rename file</td>
<td></td>
</tr>
<tr>
<td>#1100 Read specification of file</td>
<td>2</td>
</tr>
<tr>
<td>#1400 Describe file area holding given bucket</td>
<td></td>
</tr>
<tr>
<td>#1500 Extend file into specified area</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = Action successful</td>
<td>4</td>
</tr>
<tr>
<td>1 = Action not successful</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Bit 0 = 0</td>
<td></td>
</tr>
<tr>
<td>Mode #500:</td>
<td></td>
</tr>
<tr>
<td>Mode #700:</td>
<td></td>
</tr>
<tr>
<td>Mode #1100, #1400, #1500:</td>
<td></td>
</tr>
<tr>
<td>Bit 0 = 1</td>
<td></td>
</tr>
<tr>
<td>A number identifying the cause of the failure.</td>
<td>5</td>
</tr>
<tr>
<td>See Direct Access Manual</td>
<td></td>
</tr>
<tr>
<td>Modes #500, #1400:</td>
<td>6</td>
</tr>
<tr>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>Mode #700:</td>
<td></td>
</tr>
<tr>
<td>New file name</td>
<td></td>
</tr>
<tr>
<td>Mode #1500:</td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Bits 1 to 23</td>
<td></td>
</tr>
<tr>
<td>Start cylinder number of area</td>
<td></td>
</tr>
<tr>
<td>Mode #500:</td>
<td>7</td>
</tr>
<tr>
<td>Number of blocks by which file is to be extended or contracted.</td>
<td></td>
</tr>
<tr>
<td>Mode #700:</td>
<td></td>
</tr>
<tr>
<td>As Word 2</td>
<td></td>
</tr>
<tr>
<td>Mode #1400:</td>
<td></td>
</tr>
<tr>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>Mode #1500:</td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Bits 1 to 23</td>
<td></td>
</tr>
<tr>
<td>End cylinder number of area</td>
<td></td>
</tr>
<tr>
<td>Mode #500:</td>
<td>8</td>
</tr>
<tr>
<td>Serial number of storage unit onto which file is to be extended.</td>
<td></td>
</tr>
<tr>
<td>Mode #700:</td>
<td></td>
</tr>
<tr>
<td>As Word 2</td>
<td></td>
</tr>
<tr>
<td>Mode #1400:</td>
<td></td>
</tr>
<tr>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>Mode #500:</td>
<td>9</td>
</tr>
<tr>
<td>Word 4 may be set as follows</td>
<td></td>
</tr>
<tr>
<td>Zero — extension only on current storage unit</td>
<td></td>
</tr>
</tbody>
</table>

4322(7.72) Appendix 4 33
<table>
<thead>
<tr>
<th>Control area</th>
<th>Basic Direct Access Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 5</td>
<td>Mode #700: Zero</td>
</tr>
<tr>
<td>Word 6</td>
<td>Mode #700:</td>
</tr>
<tr>
<td></td>
<td>Bits 0 to 11: New generation number</td>
</tr>
<tr>
<td></td>
<td>Bits 12 to 23: New version number</td>
</tr>
<tr>
<td>Word 7</td>
<td></td>
</tr>
<tr>
<td>Unified Direct Access Standards</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Mode #1500: Start segment number of area required.</td>
<td>Positive — extension only on storage unit specified. Negative — if no room on current unit, extension may take place on any on-line storage unit.</td>
</tr>
<tr>
<td>Mode #700:</td>
<td>11 For mode #500, Word 4 is used as an additional reply word and gives the serial number of the storage unit on which the last file area lies.</td>
</tr>
<tr>
<td>Mode #1400:</td>
<td>12 For mode #500 only Words 0 to 4 are required in the control area.</td>
</tr>
<tr>
<td>Mode #1500: End segment number of area required</td>
<td>13 For UDAS mode #1500 only Words 0 to 5 are required in the control area.</td>
</tr>
<tr>
<td>Mode #700:</td>
<td>14 For mode #700 only Words 0 to 6 are required in the control area.</td>
</tr>
<tr>
<td>Bits 0 to 11: New generation number</td>
<td></td>
</tr>
<tr>
<td>Bits 12 to 23: New version number</td>
<td></td>
</tr>
<tr>
<td>Mode #1400: Bucket number</td>
<td></td>
</tr>
</tbody>
</table>

Appendix 4

432217.72)
### Magnetic Drum Operations

<table>
<thead>
<tr>
<th>Control Word</th>
<th>Transfer Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 0</td>
<td></td>
</tr>
<tr>
<td>Bits 0 to 8:</td>
<td>Type</td>
</tr>
<tr>
<td>Bits 9 to 23:</td>
<td>Mode</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 1 Reply</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>0 = Transfer completed</td>
</tr>
<tr>
<td></td>
<td>1 (that is Reply word set negative)</td>
</tr>
<tr>
<td></td>
<td>= Transfer in progress</td>
</tr>
<tr>
<td>Bits 1, 2</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 3</td>
<td>1 = Repeated parity failure</td>
</tr>
<tr>
<td>Bits 4 to 17</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bits 18 to 23</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 2</th>
<th>Word Count. Maximum is drum file size or program limit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Word 3</th>
<th>Word address in core store if start of data</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Word 4</th>
<th>Word address, relative to start of file, of drum area for transfer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Word 5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open and Close Operations</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>#100 Open named file in overlay mode</td>
<td>1. If 256 is added to the type then the X address of the order is taken to refer to an accumulator containing the file number.</td>
</tr>
<tr>
<td>#200 Open named file in input mode</td>
<td></td>
</tr>
<tr>
<td>#400 Open file in output mode and label for retention.</td>
<td></td>
</tr>
<tr>
<td>#600 Open scratch file</td>
<td></td>
</tr>
<tr>
<td>#1000 Close file</td>
<td></td>
</tr>
<tr>
<td>#1400 Close named file and abandon it.</td>
<td></td>
</tr>
</tbody>
</table>

| 0 = File successfully opened | 2. For modes #400, #600 additional bits may be set in the Reply word if Bit 0 = 1: |
| 1 = File not successfully opened | | Bits 3 to 23 Insufficient drum space available. |
| Reserved | greater than Amount available is given in Bits 3 |
| Reserved | 1023 — 23 |
| Reserved | |
| Bit 0 = 0 Indeterminate | Bits 6 to 23 |
| Bit 0 = 1 A number identifying the cause of the failure. See Direct Access Manual. | |

| Modes #100, #200, #400: Filename | 3. The reply word is indeterminate for close modes #1000, #1400. |
| Mode #600: Reserved | |
| Modes #1000, #1400: Ignored | |

| As word 2 | 4. For close modes #1000, #1400 only Words 0 to 3 are requested in the control area |
| As word 2 | |

| Modes #100, #200: Zero | 5. For transfer modes 0, 1 only Words 0 to 4 are required in the control area |
| Modes #400, #600: File size in words. Must be a multiple of 1024. | 6. For modes #100, *200 on successful completion, Word 5 will be set to contain the file size in words. |
Appendix 5  Example Compiler Listings

The following pages illustrate:

- a PLAN 3 compiler listing;
- a full PLAN 3 compiler listing;
- a PLAN 1 compiler listing.

For an illustration of a store map reference should be made to the later parts of the full PLAN 3 compiler listing illustration.

For an explanation of these printouts, see Chapter 9.

Compilation of the program #CTMT is used in the PLAN 3 illustrations. A description of this program follows.

PLAN 3 PROGRAM # CTMT (Card to Magnetic Tape Transcription Program using the I/O Generator and M.T.H. Systems)

The program transfers data from 80-column punched cards to magnetic tape that conforms to the ICL 1900 series tape-organization standards.

The M.T.H. system has been used at the physical level of file control, each block being one record made up of data from a number of cards preceded by one word containing the record size in binary. The number of cards in each record (maximum 25) is determined by a given parameter count.

If there are insufficient data cards to fill the last data block completely, this block is space-filled.

Input Data

The punched card input pack consists of a parameter card (described below), data cards, an end marker card, and a blank card. The data cards may contain any alphanumeric or symbolic data distinguishable from the end marker card, punched in the ICL 64-Character Card Code. The end marker card has an asterisk (*) punched in columns 1 to 4 inclusive.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Parameter Card Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
<td>WRI,</td>
</tr>
<tr>
<td>5 to 16</td>
<td>12-character file name</td>
</tr>
</tbody>
</table>
| 17      | Variable: generation number followed by a comma (maximum seven characters)
|        | Variable: retention period followed by a comma (maximum four characters)
|        | Variable: number of cards per record followed by a comma (maximum two characters)

An example of the punching on the parameter card is:

WRI, CTMTTESTFILE 1,21,5,

Output Format of Magnetic Tape

- Header Label
- Start of Data Sentinel
- Data Blocks
- Trailer Label

The parameter card and end marker card are not recorded on the tape.
The Compiler Printouts

The compiler listing for #CTMT is shown on pages 29 to 35, and the full compiler listing is shown on pages 36 to 44. It will be noticed that these compilations were not error free.
<table>
<thead>
<tr>
<th>0001</th>
<th>#PROGRAM</th>
<th>CTMT70/S86A</th>
<th>CTM0010</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>#LOWER</td>
<td>BUFFA(1002)</td>
<td>CTM0020</td>
</tr>
<tr>
<td>0003</td>
<td>#LOWER</td>
<td>3/0,0,60,0/BUFFA</td>
<td>CTM0030</td>
</tr>
<tr>
<td>0004</td>
<td>#ROCRD</td>
<td>3/0,0,60,0/BUFFA</td>
<td>CTM0040</td>
</tr>
<tr>
<td>0005</td>
<td>#ROCRD</td>
<td>3/0,0,60,0/BUFFA</td>
<td>CTM0050</td>
</tr>
<tr>
<td>0006</td>
<td>PAREX</td>
<td>24HINCORRECT PARAMETER CARD</td>
<td>CTM0060</td>
</tr>
<tr>
<td>0007</td>
<td>HESS</td>
<td>24/PAREX</td>
<td>CTM0070</td>
</tr>
<tr>
<td>0008</td>
<td>MESS</td>
<td>#40077777</td>
<td>CTM0080</td>
</tr>
<tr>
<td>0009</td>
<td>CADDL</td>
<td>25</td>
<td>CTM0090</td>
</tr>
<tr>
<td>0010</td>
<td>BLKSZ</td>
<td>501</td>
<td>CTM0100</td>
</tr>
<tr>
<td>0011</td>
<td>TAPE</td>
<td>261/#400,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>CTM0110</td>
</tr>
<tr>
<td>0012</td>
<td></td>
<td>0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>CTM0120</td>
</tr>
<tr>
<td>0013</td>
<td>#PROGRAM</td>
<td>ALLOT CARD READER</td>
<td>CTM0130</td>
</tr>
<tr>
<td>0014</td>
<td>#ENTRY</td>
<td>ALLOT</td>
<td>CTM0140</td>
</tr>
<tr>
<td>0015</td>
<td>START</td>
<td>CRO</td>
<td>CTM0150</td>
</tr>
<tr>
<td>0016</td>
<td>LDX</td>
<td>0</td>
<td>CTM0160</td>
</tr>
<tr>
<td>0017</td>
<td>GPZ</td>
<td>0</td>
<td>CTM0170</td>
</tr>
<tr>
<td>0018</td>
<td>SUSWT</td>
<td><strong>+</strong></td>
<td>CTM0180</td>
</tr>
<tr>
<td>0019</td>
<td>BRN</td>
<td>START</td>
<td>CTM0190</td>
</tr>
<tr>
<td>0020</td>
<td>PERI</td>
<td>#ROCRD</td>
<td>CTM0200</td>
</tr>
<tr>
<td>0021</td>
<td>SUSBY</td>
<td>0</td>
<td>CTM0210</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0150</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0170</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0180</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0190</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTM0210</td>
</tr>
<tr>
<td>Line</td>
<td>Code</td>
<td>Comment</td>
<td>Location</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>0022</td>
<td>LOX 4</td>
<td>BUFFA</td>
<td>CTM0220</td>
</tr>
<tr>
<td>0023</td>
<td>BXE 4</td>
<td>'4HWR', 1, 2</td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td>SUSTY</td>
<td>MESS</td>
<td>CTM0230</td>
</tr>
<tr>
<td>0025</td>
<td>LDCT 2</td>
<td>12</td>
<td>CTM0240</td>
</tr>
<tr>
<td>0026</td>
<td>LOCH 5</td>
<td>BUFFA+1(2)</td>
<td>CTM0250</td>
</tr>
<tr>
<td>0027</td>
<td>DCH 8</td>
<td>TAPE+2(2)</td>
<td>CTM0260</td>
</tr>
<tr>
<td>0028</td>
<td>BCRX 2</td>
<td>'2'</td>
<td>CTM0270</td>
</tr>
<tr>
<td>0029</td>
<td>LDX 2</td>
<td>'T/BUFFA+4,0'</td>
<td>CTM0280</td>
</tr>
<tr>
<td>0030</td>
<td>CALL 0</td>
<td>DECAL</td>
<td>CTM0290</td>
</tr>
<tr>
<td>0031</td>
<td>STO 5</td>
<td>TAPE+6</td>
<td>CTM0300</td>
</tr>
<tr>
<td>0032</td>
<td>ANDX 2</td>
<td>MASK</td>
<td>CTM0310</td>
</tr>
<tr>
<td>0033</td>
<td>LOCT 3</td>
<td>4</td>
<td>CTM0320</td>
</tr>
<tr>
<td>0034</td>
<td>ADX 2</td>
<td>3</td>
<td>CTM0330</td>
</tr>
<tr>
<td>0035</td>
<td>CALL 0</td>
<td>DECAL</td>
<td>CTM0340</td>
</tr>
<tr>
<td>0036</td>
<td>STO 5</td>
<td>TAPE+7</td>
<td>CTM0350</td>
</tr>
<tr>
<td>0037</td>
<td>#</td>
<td>SHOULD TEST THAT RETENTION PERIOD NOT GREATER THAN 4095</td>
<td>CTM0360</td>
</tr>
<tr>
<td>0038</td>
<td>ANDX 2</td>
<td>MASK</td>
<td>CTM0370</td>
</tr>
<tr>
<td>0039</td>
<td>LDX 2</td>
<td>'2000000'</td>
<td>CTM0380</td>
</tr>
<tr>
<td>0040</td>
<td>CALL 0</td>
<td>DECAL</td>
<td>CTM0390</td>
</tr>
<tr>
<td>0041</td>
<td>STO 5</td>
<td>CRD8L</td>
<td>CTM0400</td>
</tr>
<tr>
<td>0042</td>
<td>#</td>
<td>SHOULD TEST RECORD LENGTH NOT GREATER THAN 23 CARDS</td>
<td>CTM0410</td>
</tr>
<tr>
<td>0043</td>
<td>LDW 6</td>
<td>1</td>
<td>CTM0420</td>
</tr>
<tr>
<td>0044</td>
<td>MPA 5</td>
<td>'20'</td>
<td>CTM0430</td>
</tr>
<tr>
<td>0045</td>
<td>STO 6</td>
<td>BLKSZ</td>
<td>CTM0440</td>
</tr>
<tr>
<td>0046</td>
<td>MTEF 0</td>
<td>0,0, TAPE, ERROR</td>
<td>CTM0450</td>
</tr>
<tr>
<td>0047</td>
<td>LOX 7</td>
<td>BLKSZ</td>
<td>CTM0460</td>
</tr>
<tr>
<td>0048</td>
<td>STOR 7</td>
<td>BUFFA</td>
<td>CTM0470</td>
</tr>
<tr>
<td>0049</td>
<td>LD 2</td>
<td>CNRL</td>
<td>CTM0480</td>
</tr>
<tr>
<td>0050</td>
<td>LOCT 7</td>
<td>0(2)</td>
<td>CTM0490</td>
</tr>
<tr>
<td>0051</td>
<td>STO 1</td>
<td>1</td>
<td>CTM0500</td>
</tr>
<tr>
<td>0052</td>
<td>CARD INPUT</td>
<td>CRO, ENDCD</td>
<td>CTM0510</td>
</tr>
<tr>
<td>0053</td>
<td>INDIS</td>
<td>CRO</td>
<td>CTM0520</td>
</tr>
<tr>
<td>0054</td>
<td>A</td>
<td>80/BUFFA+1(1), 80</td>
<td>CTM0530</td>
</tr>
<tr>
<td>0055</td>
<td>END 7</td>
<td>++2</td>
<td>CTM0540</td>
</tr>
<tr>
<td>0056</td>
<td>BXU 7</td>
<td>++2</td>
<td>CTM0550</td>
</tr>
<tr>
<td>0057</td>
<td>BRN 1</td>
<td>FIRST</td>
<td>CTM0560</td>
</tr>
<tr>
<td>0058</td>
<td>ADN 1</td>
<td>20</td>
<td>CTM0570</td>
</tr>
<tr>
<td>0059</td>
<td>BRN 1</td>
<td>CARP</td>
<td>CTM0580</td>
</tr>
<tr>
<td>0060</td>
<td>FIRST</td>
<td>LOX 1</td>
<td>CTM0590</td>
</tr>
<tr>
<td>0061</td>
<td>STO 3</td>
<td>BLKSZ</td>
<td>CTM0600</td>
</tr>
</tbody>
</table>

---

Appendix S
0062 WRBLK MTR800 BUFFA(5),BLKSZ,5 [WRITE A BLOCK TO TAPE] CTM0620
0063 STO 1 BUFFA(1) CTM0630
0064 LDCT 7 OK2) CTM0640
0065 NEXT INPUT CRO CTM0650
0066 INDIS CRO CTM0660
0067 A 10/BUFFA(1) CTM0670
0068 END 7 #2 [ANY MORE CARDS TO FILL BLOCK] CTM0680
0069 BUX 1 [UPDATE MODIFIER] CTM0690
0070 BRN SECND CTM0700
0071 ADN 1 20 CTM0710
0072 BRN NEXT CTM0720
0073 SECND HTUS0 [CHECK TRANSFER] CTM0730
0074 ADN 1 21 CTM0740
0075 BRN 1 BLKSZ Cym750
0076 ERX 1 BLKSZ CTM0760
0077 ERX 3 BLK6Z CTM0770
0078 BRN WRBLK CTM0780
0079 ERROR SWSW 2KPF CTM0790
0080 BRN START CTM0800
0081 FILL ADN 1 20 CTM0810
0082 EDNCD BEE 1 DONE CTM0820
0083 BKE 1 BLKSZ,DONE CTM0830
0084 OUT BUFFA(1) [SPACEFILL REST OF BUFFER] CTM0840
0085 B 80 CTM0850
0086 END BUX 7 FILL CTM0860
0087 BUX 7 FILL CTM0870
0088 HTUS0 [CHECK TRANSFER] CTM0880
0089 ERX 3 BLKSZ CTM0890
1280 010 1 0 10
1281 070 1 1349
1282 00030056 12333
1283 0000000052705 13373
1284 010 1 1 45
1285 124 7 2 0
1286 010 1 0 1058
1287 070 1 1769
1288 000003445 1829
1289 010 1 0 1056
1290 070 1 1793
1291 000003550 1892
1292 070 3 1851
1293 01200056 46
1294 070 1 1893
1295 070 1 1901
1296 060 7 1208
1297 074 0 1300
1298 101 1 0 20
1299 074 0 1288
1300 010 1 0 10
1301 070 1 1359
1302 #00000000
1303 101 1 0 21
1304 003 1 0 1059
1305 074 0 1 1288
1306 022 3 0 1055
1307 074 0 1280
1308 101 0 0 3110
1309 074 0 1222
1310 101 1 0 12
1311 050 1 1350
1312 026 1 0 1085
1313 074 6 1330
1314 01C 1 0 1058
1315 070 1 1863
1316 00400056 46
1317 #000003550 1896
1318 070 3 1289
1319 #12000000
1320 070 1 1901
1321 060 7 1310
1322 010 1 0 10
1323 070 1 1359
1324 #00000000
1325 022 3 0 1085
1326 010 1 0 10
1327 070 1 1349
1328 #00030055 12333
0090  MTWRB 0  BUFFA(3),BLKSZ,5  WRITE A BLOCK  CTM0900
0091  DONE  MTSUS 0  CHECK TRANSFER  CTM0910

0092  MTEND 0  CLOSE  [CLOSE ANDREW TAPE]
0093  SUSWT 24HH  [END OF RUN]
0094  BRN  START  CTM0920
0095  DECAL  STOZ  4  [CONVERT DECIMAL TO BINARY]
0096  STOZ  5  [CONVERT NUMBER]
0097  CDI  4  0(2)
0098  GC5  **2
0099  SCHX  2  **2
0100  LOCH  6  0(2)

0101  BXE 6  '34',**2  [TEST EXIT CHAR = COMMA]
0102  SUSYT  MESS
0103  GCHX 2  **1
0104  EXIT 0  0
0105  #END

1329  #00632075  13373
1330  010  0  10
1331  070  1  1350
1332  #00632075
1333  010  0  10
1334  070  1  1375
1335  #00632075
1336  161  0  2000
1337  074  0  1222
1338  033  0  4
1339  033  0  5
1340  043  4  2
1341  074  5
1342  066  2  1340
1343  026  6  2
1344  026  6  1145
1345  074  6  1347
1346  160  0  1082
1347  066  2  1346
1348  072  0  0
PLAN 1 COMPILER LISTING

The following illustration shows a listing of a program #PRXT.

**PLAN(XPLP #44) LISTING**

<table>
<thead>
<tr>
<th>CR / CP</th>
<th>PROGRAM</th>
<th>Peripheral</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>PRXT/PRINT</td>
<td>LP0, CR0</td>
<td>PFCC, PRINT(30), CARD(4U)</td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0027</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0037</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0039</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0041</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0042</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0043</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0044</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0045</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0046</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0047</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0048</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0049</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0051</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0052</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0053</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0054</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0055</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0056</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0057</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0058</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0059</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**END**
Appendix 6  Overlay programming

Overlay programming is based on the principle of program segmentation, only a certain number of segments being held in store at a given time. The core store required by a program may be divided into an area of permanent storage and a number of overlay areas. The permanent storage contains, for the duration of the run, a run-time package (which calls in overlays) and at least one other segment of the user program. ICL provides a number of different overlay run-time packages: each is unique to a particular storage medium. The packages are held in the subroutine library, and the appropriate package is consolidated, having been specified under a #PERMANENT directive in the steering segment (see below). Each overlay area can hold, at any one time, one of a number of overlay units; each unit consists of one or more segments. Hence, although a number of overlay units may be present in store at the same time, they must have been specified as being in different overlay areas. Overlay organization is defined at consolidation time by means of a steering segment.

THE STEERING SEGMENT

Organization of PLAN segments into overlays, and the specification of the backing store medium from which the program is to be overlaid, is by means of a steering segment, which must be the first segment of the program.

The steering segment must contain at a minimum a #OVERLAY directive that specifies the overlay areas and units to which the segments belong, and a #PERMANENT directive specifying the name of the run-time overlay package required. Each run-time package is associated with a specific storage medium, and its incorporation in the program determines the medium on which the whole program is to be held (but see Transcribing an overlay program below). The other segments to be held in permanent storage may also be listed under #PERMANENT, but it is not necessary to do so.

If a preset data block is required by more than one segment within an overlay unit, but is not required by any segment in another unit, then the steering segment will contain a #LOWER or #UPPER directive that specifies the appropriate blockname as OVERCOMMON; when the compiler encounters a blockname under #LOWER COMMON/ or #UPPER COMMON/ within an ordinary segment, then if the blockname has been specified as OVERCOMMON in the steering segment, the block will be treated as common only within the unit to which the segment belongs, and will be included in the overlay and not in permanent storage; whereas a blockname encountered under #LOWER COMMON/ or #UPPER COMMON/ that has not been defined as OVERCOMMON in the steering segment, will be included in permanent storage.

Consolidation of an overlay program may be speeded by presenting the source input to the compiler in the following order: first, overlay areas in ascending sequence; within overlay areas, overlay units in ascending sequence; and permanent segments last. The #ORDER directive is entered in the steering segment to indicate to the compiler that this order has been observed; if #ORDER is not entered, the compiler will assume that the segments are not in order, and consolidation will take longer.

If an overlay program is to be compiled with a segment or segments missing, #OMIT is entered in the steering segment.

Other directives that may appear in the steering segment of an overlay program, but that are not specifically connected with overlays are: #CMODE, #ELASTIC, #ERRORSEG, #HMODE, #LIBRARY, #PMODE.

#COMPLETE must not be used, as it would prevent consolidation of the run-time overlay package.

The use of all directives is fully explained in Chapter 6.

The steering segment must not contain operation statements.

DATA ORGANIZATION

Preset

Preset data may be overlaid and will be reset each time the overlay unit containing it is brought into store.

A block required by more than one segment must be defined as COMMON in each segment that requires it. It will
be placed in permanent storage unless it has been specified as OVERCOMMON in the steering segment and all the segments in which it is defined are within the same overlay unit; in the latter case it will be overlaid, and will be reset each time the unit is brought into store. See Chapter 6, #LOWER and #UPPER directives, for further details.

Variable

Variable data may not be overlaid. Variable data areas defined in overlaid segments will be placed in permanent storage, and will be available at all times to segments defining them. Thus an overlaid segment may save variable data from one call-time to another.

The amount of permanent storage may therefore be reduced if working areas whose contents need not be saved from one call-time to the next are defined as preset, not as variable. This can be ensured by defining the field under a #LOWER or #UPPER directive under which preset fields are also declared; if the compiler finds both preset and variable data statements under the same #LOWER or #UPPER directive statement, then the whole area is put into preset.

BALANCING

Balancing is the technique of rearranging an overlay program so as to minimise its core usage.

Magnetic tape and cassette tape overlays

The system overlays by relativizer. Just as in a non-overlay program, the total core area allocated to the program is divided into categories, and the start address of the area of core store required for each category is identified by a relativizer (e.g. LP for lower preset). An overlay area is not one continuous area of storage, but is divided among the areas for the different categories of storage: for example, if an overlay program consisted of overlay area 1, overlay area 2, and permanently stored segments, then the lower preset area of storage (associated with relativizer LP) might contain:

1. Lower presets for overlay area 1
2. Lower presets for overlay area 2
3. Lower presets for permanent segments

and the program operations area of storage (associated with relativizer PR) would contain:

1. Program operations for permanently stored segments
2. Program operations for overlay area 1
3. Program operations for overlay area 2

and so on. (Note that, except in the case of program operations, portions of overlay areas precede portions of permanent segments, and that the portions of overlay areas are in area number order.) Within the area for a category, the start address of the portion of each overlay area is fixed, regardless of which units are in store (for example, the first word of LP for area 2 is fixed, regardless of whether area 1 currently holds unit 1, or unit 2, or unit 3).

The amount of store in a given category allocated to an overlay area is therefore the maximum amount required for that category by any overlay unit but may be held in the overlay area. The overlay area's core requirement is therefore minimised if all the overlay units which may be held in the area use the same amount of storage in each category. For example, suppose that area 1 is the overlay area in which overlay units 1, 2 and 3 may be held, and these units have the following core requirements:

<table>
<thead>
<tr>
<th>Unit 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower preset</td>
<td>430 words</td>
</tr>
<tr>
<td>Literals</td>
<td>80 words</td>
</tr>
<tr>
<td>Upper preset</td>
<td>260 words</td>
</tr>
<tr>
<td>Program instructions</td>
<td>512 words</td>
</tr>
<tr>
<td>Total for unit 1</td>
<td>1282 words</td>
</tr>
</tbody>
</table>
Unit 2
Lower preset 500 words
Literals 0 words
Upper preset 250 words
Program instructions 521 words
Total for unit 2 1271 words

Unit 3
Lower preset 249 words
Literals 8 words
Upper preset 541 words
Program instructions 472 words
Total for unit 3 1270 words

The requirement for area 1 can be calculated as follows:

<table>
<thead>
<tr>
<th>Unit 1</th>
<th>Unit 2</th>
<th>Unit 3</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>words</td>
<td>words</td>
<td>words</td>
</tr>
<tr>
<td>Lower preset</td>
<td>430</td>
<td>500</td>
<td>249</td>
</tr>
<tr>
<td>Literals</td>
<td>80</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Upper preset</td>
<td>260</td>
<td>250</td>
<td>541</td>
</tr>
<tr>
<td>Program instructions</td>
<td>512</td>
<td>521</td>
<td>472</td>
</tr>
</tbody>
</table>

Total for area 1 1642

This requirement of 1642 words could be reduced to 1282 words (the size of the largest unit to be held in area 1) if the usage of the particular units were suitably manipulated between categories.

1 REPLACEMENT OF LITERALS BY PRESET. 80 words are required for literals, but unit 2 uses none of these and unit 3 uses only 8. 500 words are reserved for lower presets, but unit 1 uses only 430 of these and unit 3 uses only 249. 70 words can therefore be saved by replacing all literals by presets.

2 REARRANGEMENT OF PRESET. 500 words are reserved for lower preset and 541 for upper preset; yet no unit uses more than 790 words of presets altogether. Thus 251 words could be saved by moving preset data in unit 3 from upper to lower (provided this does not cause the total area of lower storage to exceed 4096 words).

3 TABLES IN PROGRAM AREA. The changes so far suggested would allow 321 words to be saved. This leaves a discrepancy of 39 words, which could be further reduced if tables from upper preset can be moved to the program operations area. This can be done by declaring them under #CUE; the user must ensure that the program will never attempt to execute these “instructions”. In this example 10 words might be moved from upper preset to program in unit 1, and 40 words in unit 3.

The degree to which an ideal solution can be approximated depends upon the particular circumstances: for example locations which were contiguous in the original arrangement of a unit may have to be kept together in the balanced overlay. However, an ideal balancing of the example overlay area described above might be as follows:

<table>
<thead>
<tr>
<th>Unit 1</th>
<th>Unit 2</th>
<th>Unit 3</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower preset</td>
<td>510</td>
<td>500</td>
<td>510</td>
</tr>
<tr>
<td>Literals</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Upper preset</td>
<td>250</td>
<td>250</td>
<td>248</td>
</tr>
<tr>
<td>Program instructions</td>
<td>522</td>
<td>521</td>
<td>512</td>
</tr>
<tr>
<td>Total</td>
<td>1282</td>
<td>1271</td>
<td>1270</td>
</tr>
</tbody>
</table>

In general, balancing will be easier, the fewer relativizers are used.
Store analysis facilities for programs overlaid from magnetic tape are provided by the program #XPSU (see PLAN Program Development Aids, 1st edition, T.P. number 4179, pages 37 to 45, for details of this program, and an example).

The possibilities of reducing core usage by means of the technique discussed above are clearly limited by the sizes of the units. If an area is to hold a unit of 1024 words and a unit of 2048 words, 1024 words will be wasted when the smaller unit is in store, regardless of the arrangement of storage between categories. In such cases balancing will require a rearrangement of the program’s overlay structure.

**Direct access overlays**

The user should consult the current User Notices for details of balancing in programs overlaid from direct access backing store.

**UNIT NUMBERS**

Logical unit number 0 of the peripheral type from which the program is overlaid is reserved for use by the overlay system, and should not be used by the overlaid object program.

**OVERLAY PSEUDO-OPERATIONS**

The following PLAN pseudo-operations are provided for use in overlay programs: ENTER, RECAL and BRING. Each of these generates a branch to the run-time overlay package. In each case the user specifies in the operand field the name of a cue within an overlay unit. ENTER causes the run-time overlay package to ascertain which overlay unit contains the specified cue: if this unit is already in core store, the program then branches to a specified location within it; otherwise it brings the unit into store before branching. RECAL is similar, except that it causes the program to bring into store the unit containing the specified cue, even if it is already in core: this enables the programmer to reset a unit to its initial condition after it has been altered in some way. BRING causes the program to bring into core the unit containing the specified cue (unless it is in core store already), but not to enter it. These pseudo-operations are described in Chapter 5, pages 55 to 58.

**RUN-TIME OVERLAY PACKAGES**

The following overlay packages are available. Each controls the bringing in of overlays from the specified medium.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose and characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>%EROL</td>
<td>For overlays held on E.D.S., Twin-E.D.S. or F.D.S.</td>
</tr>
<tr>
<td>%TROL</td>
<td>For overlays held on magnetic tape (but see %AATROL below).</td>
</tr>
<tr>
<td>%AATROL</td>
<td>For overlays held on magnetic tape when a real time device is in use.</td>
</tr>
<tr>
<td></td>
<td>If a program running a real time device in a subprogram is to be overlaid from magnetic tape, the overlay package specified under the #PERMANENT directive must be %AATROL rather than %TROL. This requirement arises because the overlaying process used by %TROL cannot be interrupted by a higher priority member.</td>
</tr>
<tr>
<td></td>
<td>When using this system, all of the ENTER, RECAL and BRING pseudo-instructions must be performed in member 0, and, if any other member is being overlaid, it must be suspended. Overlaying of a member other than member 0 should be achieved only by BRING, never by ENTER or RECAL.</td>
</tr>
<tr>
<td>%CROL</td>
<td>For overlays held on cassette tape.</td>
</tr>
<tr>
<td>%DROL</td>
<td>For overlays held on a magnetic drum. A binary dump of the overlay program will be produced on a magnetic tape in the same way as for a program to be overlaid from magnetic tape. When the program is to be run, a normal Find message should be input. The permanent part of the program is loaded into core, and then a drum file is created to which the overlays are copied. The magnetic tape holding the program is released, and the program halts with the message:</td>
</tr>
<tr>
<td></td>
<td>HALTED:—LD</td>
</tr>
<tr>
<td></td>
<td>It then can be entered as specified by the programmer.</td>
</tr>
</tbody>
</table>
Purpose and characteristics

For overlays held on disc or in core.

The system is intended for use on large machines for large programs which do not use 22-bit addressing mode and extended branch mode facilities. Such programs may be overlaid so that they do not occupy more than 32K of core store; the disc/core overlay system allows additional core store to be used to speed up the process of overlay changing. Proven programs can thus be made to take advantage of an increase in core store availability merely by recompiling; no change in the overlay structure is needed.

The basic principle of the system is that some overlays are held in an area above (but belonging to) the object program, and the remainder on a disc file. The user can specify which overlays are to be held in core, and which on disc backing store. When an overlay is required, it is either shifted down to its run-time position (if it is an overlay held in core), or brought in from disc backing store (if it is an overlay held on backing store). Shifting an overlay in core store is faster than bringing one from backing store, and hence the disc/core overlay system is likely to be most advantageous when:

1. the program uses frequent overlay changes,
2. the overlays most often changed are held in core store, and
3. there is sufficient core in the processor for the excess core used by the overlay system not to inconvenience the running of other programs.

The system consists of two run-time packages, %FINROL and %AAINCORE.

A program to be overlaid from disc/core should have the usual overlay steering segment, which should specify %FINROL under the #PERMANENT directive. It is not necessary to specify %AAINCORE, but %AAINCORE must occur after %FINROL in the subroutine library.

The user controls whether particular overlays are held in core or left on disc by choosing appropriate unit numbers for his overlay units. All overlay units in the range 1 to 500 will be held in core, and all those from 501 to 1023 on disc. This division applies to each overlay area.

The object program is assembled in the same way as for any other disc overlaid program. The binary program must be held in a disc file. When the program is loaded, additional core store will be allocated automatically, and the required overlays will be transferred from the disc to the additional core store. The disc file will normally remain assigned to the object program so that the overlays not held in core store may be accessed. However, if the user has indicated that all of the overlays are to be held in core store, then the file will be closed. When the overlays have been transferred, the message HALTED:-LD will be output. The user may then enter his program as required, and the pseudo-operations ENTER, RECAL and BRING will act in the normal way.

The program must not attempt to vary its core store allocation by a GIVE instruction with N(M) = 4, nor should it use any data areas other than those actually requested in the object program. In particular, the reply to a GIVE instruction with N(M) = 3 must not be taken to mean that the object program may use that amount of core.

In a multiprogramming environment, the process of transferring the overlays from backing store to core store may be speeded up by allocating to the object program in the FIND message the eventual core requirement of the system. This figure will be known as soon as the program has been run for the first time. The saving consists of the time required to do one or two GIVE N(M) = 4 instructions.

The program may run in any legal mode combination.

An alternative system for overlays held on disc or in core.

This system is similar in operation to the standard system (%FINROL) described above but gives increased efficiency of overlaying. A full description of %FINROLX will be found in the 1900 Series manual Compiling Systems, Edition 1, TP 4241.

CONSOLIDATION AND LOADING

See Chapter 7, pages 57 and 58.
ORDER OF RELATIVIZERS

In 15AM, DBM overlay programs, the program instruction area is at the top of the store (that is, UP precedes PR). (Note: it is not guaranteed that all consolidators will observe this rule in future.)

TRANSCRIBING AN OVERLAY PROGRAM

An overlay program can be transcribed from one medium to another for storage or security purposes using the appropriate standard library manipulation utility program. When the overlay program is run, it must be held on the medium appropriate to the overlay run-time package which it contains.

OVERLAY ERROR MESSAGES

The following messages may be output by the run-time overlay package to indicate errors arising in the overlay system.

<table>
<thead>
<tr>
<th>Message</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALTED:-- OVRL ER1</td>
<td>1  The PLAN pseudo-instruction ENTER, RECAL or BRING has been referenced in an OBEEY instruction, or</td>
</tr>
<tr>
<td></td>
<td>2  The overlay calling sequence or one of the data areas belonging to the overlay subroutine has been corrupted by the program, or</td>
</tr>
<tr>
<td></td>
<td>3  The overlay magnetic tape or cassette tape has been moved by the program.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER2</td>
<td>As (2) and (3), in message OVRL ER1 above.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER3</td>
<td>If using the drum overlay subroutine %DROL: there is insufficient space on the drum to hold the overlays.</td>
</tr>
<tr>
<td></td>
<td>If using any other overlay subroutine: a format error has been detected in the overlay program.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER4</td>
<td>Format error.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER5</td>
<td>A peripheral transfer has failed while bringing down an overlay.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER6</td>
<td>Format error.</td>
</tr>
<tr>
<td>HALTED:-- OVRL ER7</td>
<td>Certain necessary information has not been set up during loading of the program from the disc. This is usually caused by use of an out-of-date version of #XPFO or #XPEO.</td>
</tr>
<tr>
<td>HALTED:-- OVRL PROG - DRUM PARITY ERROR</td>
<td>A drum parity error has been detected by the drum overlay subroutine.</td>
</tr>
<tr>
<td>HALTED:-- TP</td>
<td>A peripheral transfer failure has occurred while using %AATROL.</td>
</tr>
<tr>
<td>HALTED:-- E1 (#2400 4501)</td>
<td>These messages, output by the cassette tape overlay subroutine %CROL, correspond to OVRL ER1 and OVRL ER2 above.</td>
</tr>
<tr>
<td>HALTED:-- E2 (#2400 4502)</td>
<td>These messages, output by the cassette tape overlay subroutine %CROL, correspond to OVRL ER1 and OVRL ER2 above.</td>
</tr>
</tbody>
</table>

In addition to the above messages, Executive format error messages (similar to those output when loading a binary program) may be output when using the magnetic tape or cassette tape overlay subroutines %TROL or %CROL.
Appendix 7  A Summary of PLAN Compiler Facilities
NAME
#XPLD

HARDWARE REQUIREMENTS
5888 words of core store
1 paper tape reader or 1 card reader
1 paper tape punch or 1 card punch
1 line printer

DESCRIPTION
PLAN 2, single program, non-overlay
8K minimum processor size

Input
One or more source segments on paper tape or cards headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Unconsolidated semi-compiled object on paper tape or cards. A consolidated semi-compiled object program can be obtained by a separated run using #XPCA for object on paper tape or #XPCC for object on cards
LISTING On line printer

CONTROL
Entry points
(see switch settings)
20 to activate compiler

Switch settings
21 if LIST or SHORTLIST and listing required on 96 print position printer
22 if OBJECT and semi-compiled object required on cards
23 if input on cards

Input parameters
None

Steering line facilities
LIST, SHORTLIST, OBJECT
<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR0 or TR0</td>
<td>source input</td>
<td>The reader is assigned at the beginning of the run and released when a #FINISH or #SWITCH directive is read.</td>
</tr>
<tr>
<td>CP0 or TP0</td>
<td>object output</td>
<td>The punch is assigned when a steering line OBJECT is read and released when a #FINISH directive is read.</td>
</tr>
<tr>
<td>LP0</td>
<td>listing</td>
<td>The line printer is assigned when a steering line LIST or SHORTLIST is read and released when a #FINISH directive is read.</td>
</tr>
</tbody>
</table>
NAME
#XPLE

HARDWARE REQUIREMENTS
11392 words of core store
1 paper tape reader or 1 card reader
1 paper tape punch or 1 card punch
1 line printer or 1 tape punch

DESCRIPTION
PLAN 3, single program, non-overlay
16K minimum processor size

Input
One or more source segments on paper tape or cards, headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Unconsolidated semi-compiled on paper tape or cards. A consolidated semi-compiled object program can be obtained by a separate run using #XPCA for object on paper tape or #XPCC for object on cards
LISTING On line printer or paper tape

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
20 if LIST or SHORTLIST and listing required on paper tape
21 if LIST or SHORTLIST and listing required on 96 print position printer
22 if OBJECT and semi-compiled object program required on cards

Input parameters
None

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE

USE OF PERIPHERALS
CR0 or TR0 source input The reader is assigned at the beginning of the run and released when a #FINISH or #SWITCH directive is read.
<table>
<thead>
<tr>
<th>CP0 or TP0</th>
<th>object output</th>
<th>The punch is assigned when a steering line OBJECT is read, and released when a #FINISH directive is read.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP0 or TP1</td>
<td>listing and error messages</td>
<td>The device is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLF

HARDWARE REQUIREMENTS
13056 words of core store
1 paper tape reader or 1 card reader
1 line printer
1 E.D.S. or Twin E.D.S.
1 or 2 E.D.S., Twin E.D.S. or F.D.S. (optional)

DESCRIPTION
PLAN 3, batch, overlay
16K minimum processor size

Input
One or more source programs on paper tape or cards, terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Unconsolidated semi-compiled on E.D.S., Twin E.D.S. or F.D.S. in subfile format. A consolidated semi-compiled object program on disc can be obtained by a separate run using #XPCK or by parameter calling #XPCK at the end of the run.
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional OUT, OUTE, OUTF, APP, APPF, REN, WORK, NEXT, BIN, BINE, BINF, LIB, LIBE, LIBF, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST

USE OF PERIPHERALS
CRO or TP0 parameters and source input

The reader is assigned at the beginning of the run and released when a #STOP or #SWITCH directive is read.

4322(7.72) Appendix 7 71
| ED0       | overlays | The file is assigned when \#XPLF is loaded into store and released when the compiler is deleted. |
| ED2 or FD2 | unconsolidated object output (optional) | The file is assigned when an OUT, OUTE, OUTF, APP, APPE or APFF parameter is used and released at the end of the run. |
| LP0       | listing and error messages | The line printer is assigned at the beginning of the run and released at the end of the run. |
NAME
#XPLG

HARDWARE REQUIREMENTS
11456 words of core store
1 paper tape reader or 1 card reader
1 line printer
1 magnetic tape deck
2 magnetic tape decks (optional)

DESCRIPTION
PLAN 3, batch, overlay
16K minimum processor size

Input
One or more source programs on paper tape or cards terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM  Consolidated semi-compiled on magnetic tape in subfile format
LISTING  On line printer

CONTROL
Entry points
20  for paper tape
21  for cards

Switch settings
18  if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21  if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional  OUT, REN, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST, MAP(OFF), FULLLIST(OFF), BINARY

USE OF PERIPHERALS
CR0 or TR0  parameters and source input
LP0  listing and error messages

The reader is assigned at the beginning of the run and released when a #STOP or #SWITCH directive is read.
The line printer is assigned at the beginning of the run and released at the end of the run.
| MT0 | overlays | The magnetic tape is assigned when #XPLG is loaded into store and released when the compiler is deleted. |
| MT2 | work file (optional) | The magnetic tape is assigned when a steering line OBJECT is read, and released at the end of the run. |
| MT3 | object output (optional) | The magnetic tape is assigned when a steering line OBJECT is read, and released at the end of the run. |
NAME
#XPLH

HARDWARE REQUIREMENTS

$824 words of core store
1 paper tape reader or 1 card reader
1 line printer
2 magnetic tape decks
1 magnetic tape deck (optional)

DESCRIPTION

PLAN 3, batch, overlay
8K minimum processor size

Input

One or more source programs on paper tape or cards terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output

OBJECT PROGRAM Consolidated semi-compiled on magnetic tape in subfile format
LISTING On line printer

CONTROL

Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional OUT, REN, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST, MAP(OFF), FULLLIST(OFF), BINARY

USE OF PERIPHERALS

CR0 or TR0 parameters and source input
LP0 listing and error messages

The reader is assigned at the beginning of the run and released when a #STOP or #SWITCH directive is read.
The line printer is assigned at the beginning of the run and released at the end of the run.
<table>
<thead>
<tr>
<th>MT0</th>
<th>overlays</th>
<th>The magnetic tape is assigned when #XPHL is loaded into store and released when the compiler is deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT2</td>
<td>work file (optional)</td>
<td>The magnetic tape is assigned when a steering line OBJECT is read and released at the end of the run.</td>
</tr>
<tr>
<td>MT3</td>
<td>work file then object output</td>
<td>The magnetic tape is assigned at the beginning of the run and if a steering line OBJECT is read MT3 receives the object program and is released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLL

HARDWARE REQUIREMENTS
5888 words of core store
1 paper tape reader or 1 card reader
1 line printer
1 magnetic tape deck
1 magnetic tape deck (optional)
2 E.D.S. or Twin E.D.S.
1 E.D.S. or Twin E.D.S. (optional)

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source programs on paper tape or cards terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Consolidated semi-compiled on magnetic tape in subfile format
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional OUT, REN, WORK, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST, MAP(OFF), FULLLIST(OFF), BINARY
<table>
<thead>
<tr>
<th>USE OF PERIPHERALS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR0 or TR0</td>
<td>parameters and source input</td>
</tr>
<tr>
<td>EDO</td>
<td>overlays</td>
</tr>
<tr>
<td>ED1</td>
<td>scratch file</td>
</tr>
<tr>
<td>ED2</td>
<td>scratch file (optional)</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
</tr>
<tr>
<td>MT0</td>
<td>P.L.T.</td>
</tr>
<tr>
<td>MT3</td>
<td>object output (optional)</td>
</tr>
</tbody>
</table>

The reader is assigned at the beginning of the run and released when a `#STOP` or `#SWITCH` directive is read.

The file is assigned at the beginning of the run. The overlays are copied from MT0 to this device and are held there until the device is released at the end of the run.

The file is assigned at the start of the run and released at the end of the run.

The file is assigned when a steering line OBJECT is read and released at the end of the run.

The line printer is assigned at the beginning of the run and released at the end of the run.

The magnetic tape is assigned when `#XPLL` is loaded into store and released when the compiler is deleted.

The magnetic tape is assigned when a steering line OBJECT is read and released at the end of the run.
NAME
#XPLM

HARDWARE REQUIREMENTS
5824 words of core store
1 paper tape reader or 1 card reader
1 line printer
2 E.D.S. or Twin E.D.S.
1 E.D.S. or Twin E.D.S. (optional)

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source programs on paper tape or cards terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Unconsolidated semi-compiled on E.D.S. or Twin E.D.S. in subfile format. A binary program on disc can be obtained by a separate run using #XPCL or by parameter calling #XPCL at the end of the run
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional OUT, OUTE, APP, APPE, REN, WORK, NEXT, BIN, BINE, LIB, LIBE, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST

USE OF PERIPHERALS
CR0 or TR0 parameters and source input
ED0 overlays

The reader is assigned at the beginning of the run and released when a #STOP or #SWITCH directive is read.
The file is assigned when #XPLM is loaded into store and released when the compiler is deleted.
<table>
<thead>
<tr>
<th>ED1</th>
<th>scratch file</th>
<th>The file is assigned at the beginning of the run and released at the end of the run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED2</td>
<td>object output (optional)</td>
<td>The file is assigned when an OUT, OUTE, APP or APPE parameter is used and released at the end of the run.</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
<td>The line printer is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLN

HARDWARE REQUIREMENTS
17984 words of core store
1 paper tape reader and/or 1 card reader
1 line printer
1 magnetic tape deck
Up to 8 magnetic tape decks (optional)

DESCRIPTION
PLAN 4, batch, overlay
32K minimum processor size

Input
One or more source and/or semi-compiled programs on paper tape and/or cards and/or magnetic tape (written
by COSY and DISC COSY editors (#XPM, #XPMR, #XPMY, #XPMZ and #XPMQ), by GEORGE editor (#XKYA)
by any PLAN compiler or by any program with the file conforming to these specifications).

Output
OBJECT PROGRAM  Consolidated semi-compiled object on magnetic tape as composite or simple file;
unconsolidated semi-compiled segments on magnetic tape in subfile format
LISTING  On line printer

CONTROL
Entry points
20  for parameters on paper tape
21  for parameters on cards

Switch settings
None

Input parameters
*JOB, ORF, OWFA, APPA, OWFB, OWFC, APPC, RWH, WSF, SFEND, REWIND, CLOSE, ASTEER, BSTEER,
PLAN, S/C, #SWITCH, *, ****

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CONSOLIDATE, MAP, FULLIST, MAP(OFF), FULLIST(OFF),
BINARY

USE OF PERIPHERALS
CR0 or TR0  parameters  The reader is assigned at the beginning of the run and
released when a #SWITCH directive is read.
<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPO</td>
<td>listing and error messages</td>
<td>The line printer is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td>MT0</td>
<td>overlays</td>
<td>The magnetic tape is assigned when #XPLN is loaded into store and released when the compiler is deleted.</td>
</tr>
<tr>
<td>MT2</td>
<td>scratch file (optional)</td>
<td>The file is used throughout the run.</td>
</tr>
<tr>
<td>MT3</td>
<td>object output</td>
<td>This file is opened as a composite file when a parameter names the file SCRATCH TAPE and is released at the end of the run.</td>
</tr>
<tr>
<td>MT10 to MT15</td>
<td>source input or object output</td>
<td>The files are used throughout the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLP

HARDWARE REQUIREMENTS
2688 words of core store
1 paper tape reader or 1 card reader
1 paper tape punch or 1 card punch
1 line printer

DESCRIPTION
PLAN 1, single program, Main Binary/Main Overlay
4K minimum processor size

Input
OBJECT PROGRAM Unconsolidated semi-compiled on paper tape or cards. A consolidated semi-compiled object program can be obtained by a separate run using #XPCA for object on paper tape or #XPCC for object on cards
LISTING On line printer

CONTROL
Entry points
20 to activate the compiler

Switch settings
0 ON No object output
OFF Object output required
1 ON Source input on paper tape
OFF Source input on cards
2 ON Object output on paper tape
OFF Object output on cards
3 ON No listing
OFF Listing on line printer

Input parameters
None

Steering line facilities
None (performed by switch settings)
USE OF PERIPHERALS

CR0 or TR0
#XPLP main binary source input
#XPLP overlay binary

CP0 or TP0
object output

LP0
listing

The reader is assigned at the beginning of the run. #XPLP Main Binary, the source input and #XPLP Overlay Binary are loaded in turn and the device is released at the end of the run.

The punch is assigned at the beginning of the run and released at the end of the run.

The line printer is assigned at the beginning of the run and released at the end of the run.
NAME
#XPLQ

HARDWARE REQUIREMENTS
2176 words of core store
1 paper tape reader or 1 card reader
2 cassette tape stations
1 cassette tape station (optional)

DESCRIPTION
PLAN 1, single program, overlay
4K minimum processor size

Input
One or more source segments on paper tape or cards

Output
OBJECT PROGRAM  Unconsolidated semi-compiled on cassette tape in subfile format. A consolidated semi-compiled object program can be obtained by a separate run using #XPCW
LISTING  On line printer

CONTROL
Entry points
20  to activate the compiler

Switch settings
0  ON  No object output
    OFF  Object output required
1  ON  Source input on paper tape
    OFF  Source input on cards
2  ON  No listing
    OFF  Listing on line printer

Input parameters
None

Steering line facilities
None (performed by switch settings)

USE OF PERIPHERALS
CRO or TR0  source input

The reader is assigned at the beginning of the run and released at the end of the run.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT0</td>
<td>overlays</td>
<td>The cassette tape is assigned at the beginning of the run and released when the compiler is deleted.</td>
</tr>
<tr>
<td>CT1</td>
<td>scratch file</td>
<td>The cassette tape is assigned at the beginning of the run and closed at the end of the run. The cassette tape remains scratch.</td>
</tr>
<tr>
<td>CT3</td>
<td>object output (optional)</td>
<td>The cassette tape is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td>LP0</td>
<td>listing</td>
<td>The line printer is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLR

HARDWARE REQUIREMENTS
5888 words of core store
1 paper tape reader or 1 card reader (optional)
3 cassette tape stations
1 cassette tape station (optional)
1 line printer

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source and/or semi-compiled programs on cassette tape in subfile format written by COSY editor (#XPMJ or #XPMK)

Output
OBJECT PROGRAM  Unconsolidated semi-compiled on cassette tape in subfile format. A binary program can be obtained by a separate run using #XPCX
LISTING  On line printer

CONTROL
Entry points
20  for paper tape
21  for cards
22  for cassette tape (if called COMPILE FILE and no basic peripheral input device is assigned)

Switching settings
18  if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21  if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional  IN, OUT, REN, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR

USE OF PERIPHERALS
CR0 or TR0  parameters  The reader is assigned at the beginning of the run and released when the parameters have been read and checked.
<table>
<thead>
<tr>
<th>CT0</th>
<th>P.L.T.</th>
<th>The cassette tape is assigned when #XPLR is loaded into store and released when the compiler is deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT1</td>
<td>scratch file</td>
<td>The cassette tape is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td>CT2</td>
<td>object output (optional)</td>
<td>The cassette tape is assigned when an OUT parameter or the COMPILE FILE facility is used and released at the end of the run.</td>
</tr>
<tr>
<td>CT3</td>
<td>source input</td>
<td>The cassette tape is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
<td>The line printer is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLS

HARDWARE REQUIREMENTS
5888 words of core store
1 paper tape reader or 1 card reader
1 line printer
2 cassette tape stations
2 cassette stations (optional)

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source programs on paper tape or cards terminated by a #STOP directive. Each source program consists of one or more source segments headed by a steering line and terminated by a #FINISH directive.

Output
OBJECT PROGRAM Binary program on cassette tape
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional OUT, REN, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, BINARY

USE OF PERIPHERALS
CR0 or TR0 parameters and source input
CT0 P.L.T.
The reader is assigned at the beginning of the run and released when a #STOP or #SWITCH directive is read
The cassette tape is assigned when #XPLS is loaded into store and released when the compiler is deleted.
CT1  work file  
CT2  work file  
CT3  object output  
LP0  listing and error messages

The cassette tape is assigned at the beginning of the run and released at the end of the run.

The cassette tape is assigned when an OUT parameter is used and released at the end of the run.

The cassette tape is assigned when an OUT parameter is used and released at the end of the run.

The line printer is assigned at the start of the run and released at the end of the run.
NAME
#XPLT

HARDWARE REQUIREMENTS
18176 words of core store
1 paper tape reader or 1 card reader
1 line printer
One or more E.D.S. or F.D.S. files (optional)
One or more magnetic tape decks (optional)
1 paper tape reader or card reader (optional)

DESCRIPTION
PLAN 4, batch, non-overlay
32K minimum processor size

Input
One or more source and/or semi-compiled segments on paper tape and/or cards and/or E.D.S. or F.D.S. (written by DISC COSY editors (#XPMY or #XPMZ) or by a user program with the file conforming to the specification for DISC COSY) and/or magnetic tape (written by COSY or DISC COSY editors (#XPMS, #XPMR, #XPMY, #XPMZ or #XPMQ), by GEORGE editor (#XKYA), by any PLAN compiler or by any program with the file conforming to these specifications).

Output
OBJECT PROGRAM Unconsolidated semi-compiled object on E.D.S. or F.D.S. in subfile format. A consolidated semi-compiled object program can be obtained by a separate run using #XPCL
LISTING On line printer

CONTROL
Entry points
20 for parameters on paper tape
21 for parameters on cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
PROG, IN, INE, INF, INM, OUT, OUTE, OUTF, APP, APPE, APPF, REN, WSF, CLOSE, CAF, EXTE, REW,
SWITCH, STEER, SEGSTEER, PLAN, S/C, ALL, AMPL, NEXT, BIN, BINE, BINF, INC, INCE, INCF, LIB,
LIBE, LIFB, ENDPREG

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, MAP, FULLLIST

4322(7.72) Appendix 7 91
## USE OF PERIPHERALS

| CR0 or TR0 | parameters |
| ED2 or FD2 | object output (optional) |
| ED4 to ED7, FD4 to FD7, MT4 to MT7 | source input |
| LP0 | listing and error message |

The reader is assigned at the beginning of the run and released when the parameters have been read and checked.

The file is used throughout the run.

The files are opened and closed as specified by parameter.

The line printer is assigned at the beginning of the run and released at the end of the run.
NAME
#XPLV

HARDWARE REQUIREMENTS
11520 words of core store
1 paper tape reader or 1 card reader
1 line printer
2 magnetic tape decks
2 magnetic tape decks (optional)

DESCRIPTION
PLAN 3, batch, overlay
16K minimum processor size

Input
One or more source and/or semi-compiled programs on magnetic tape in subfile format written by COSY editor (#XPMR or #XPMS)

Output
OBJECT PROGRAM  Consolidated semi-compiled object on magnetic tape in subfile format
LISTING    On line printer

CONTROL
Entry points
20  for paper tape
21  for cards
22  for magnetic tape (if called COMPILcE FILE and no basic peripheral input device is assigned)

Switch settings
18  if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21  if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional   IN, OUT, REN, SUB, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, MAP, FULLLIST, MAP(OFF), FULLLIST(OFF), BINARY

USE OF PERIPHERALS
CR0 or TR0 parameters
LP0 listing and error messages

The reader is assigned at the beginning of the run and released when the parameters have been read and checked.
The line printer is assigned at the beginning of the run and released at the end of the run.
<table>
<thead>
<tr>
<th>MT0</th>
<th>overlays</th>
<th>The magnetic tape unit is assigned at the beginning of the run and released when the compiler is deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT2</td>
<td>scratch file (optional)</td>
<td>The magnetic tape unit is assigned when a steering line OBJECT is read and released at the end of the run.</td>
</tr>
<tr>
<td>MT3</td>
<td>object output (optional)</td>
<td>The magnetic tape unit is assigned when a steering line OBJECT is read and released at the end of the run.</td>
</tr>
<tr>
<td>MT4</td>
<td>source input</td>
<td>The magnetic tape unit is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLW

HARDWARE REQUIREMENTS
8888 words of core store
1 paper tape reader or 1 card reader
1 line printer
3 magnetic tape units
1 magnetic tape unit (optional)

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source and/or semi-compiled programs on magnetic tape in subfile format written by COSY editor (#XPMR or #XPMS)

Output
OBJECT PROGRAM Consolidated semi-compiled object on magnetic tape in subfile format
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards
22 for magnetic tape (if called COMPILE FILE and no basic peripheral input device is assigned)

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional IN, OUT, REN, SUB, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, MAP, FULLIST, MAP(OFF), FULLIST(OFF), BINARY

USE OF PERIPHERALS
CR0 or TR0 parameters
The reader is assigned at the beginning of the run and released when the parameters have been read and checked.

LPO listing and error messages
The line printer is assigned at the beginning of the run and released at the end of the run.
<table>
<thead>
<tr>
<th>MT0</th>
<th>overlays</th>
<th>The magnetic tape unit is assigned at the beginning of the run and released when the compiler is deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT2</td>
<td>scratch file (optional)</td>
<td>The magnetic tape unit is assigned when a steering line OBJECT is read and released at the end of the run.</td>
</tr>
<tr>
<td>MT3</td>
<td>scratch file</td>
<td>The magnetic tape unit is assigned at the beginning of the run and if a steering line OBJECT is read MT3 receives the object program and is released at the end of the run.</td>
</tr>
<tr>
<td></td>
<td>then object output</td>
<td></td>
</tr>
<tr>
<td>MT4</td>
<td>source and</td>
<td>The magnetic tape unit is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td></td>
<td>semi-compiled input</td>
<td></td>
</tr>
</tbody>
</table>
NAME
#XPLX

HARDWARE REQUIREMENTS
5888 words of core store
1 paper tape reader or 1 card reader
1 line printer
3 E.D.S. or Twin E.D.S.
1 E.D.S. or Twin E.D.S. (optional)

DESCRIPTION
PLAN 3, batch, overlay
8K minimum processor size

Input
One or more source and/or semi-compiled programs on an E.D.S. or Twin E.D.S. file written by a DISC COSY editor (#XPMY or #XPMZ) or by a sort/merge generator (#XSEG) or by a user program with the file conforming to the specification for DISC COSY.

Output
OBJECT PROGRAM Unconsolidated semi-compiled object on E.D.S. or Twin E.D.S. in subfile format. A binary program can be obtained by a separate run using #XPCL
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Mandatory IN, INE, PEND
Optional OUT, OUTE, APP, APPE, REN, WORK, SUB, NEXT, BIN, BINE, LIB, LIBE

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, MAP, FULLLIST
### USE OF PERIPHERALS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR0 or TR0</td>
<td>parameters</td>
<td>The reader is assigned at the beginning of the run and released when the parameters have been read.</td>
</tr>
<tr>
<td>ED0</td>
<td>overlays</td>
<td>The file is assigned when #XPLX is loaded into store and released when the compiler is deleted.</td>
</tr>
<tr>
<td>ED1</td>
<td>scratch file</td>
<td>The file is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
<tr>
<td>ED2</td>
<td>object output (optional)</td>
<td>The file is opened when an OUT, OUTE, APP or APPE parameter is used and is released at the end of the run.</td>
</tr>
<tr>
<td>ED4</td>
<td>source input</td>
<td>The file is assigned at the beginning of the run and is released at the end of the run.</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
<td>The line printer is assigned at the beginning of the run and released at the end of the run.</td>
</tr>
</tbody>
</table>
NAME
#XPLY

HARDWARE REQUIREMENTS
12544 words of core store
1 paper tape reader or 1 card reader
1 line printer
2 magnetic tape units
1 magnetic tape unit (optional)
1 E.D.S. or Twin E.D.S. transport
2 E.D.S. or Twin E.D.S. transports (optional)

DESCRIPTION
PLAN 3, batch, overlay
16K minimum processor size

Input
One or more source and/or semi-compiled programs on magnetic tape in subfile format written by COSY editor
(#XPMR or #XPMS) or one or more source programs on paper tape and/or cards terminated by a #STOP
directive. Each source program consists of one or more source segments headed by a steering line and terminated
by a #FINISH directive.

Output
OBJECT PROGRAM Binary program on magnetic tape
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards
22 for magnetic tape (if named COMPILE FILE and no basic peripheral input device is assigned)

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Optional IN, OUT, REN, WORK, SUB, AMEND, PEND

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, CHARGE, MAP, FULLLIST, MAP(OFF), FULLLIST(OFF), BINARY
<table>
<thead>
<tr>
<th>USE OF PERIPHERALS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CR0 or TR0</td>
<td>parameters</td>
</tr>
<tr>
<td>ED0</td>
<td>overlays and possibly scratch file</td>
</tr>
<tr>
<td>ED1</td>
<td>scratch file (optional)</td>
</tr>
<tr>
<td>ED2</td>
<td>scratch file (optional)</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
</tr>
<tr>
<td>MT0</td>
<td>P.L.T.</td>
</tr>
<tr>
<td>MT3</td>
<td>object output (optional)</td>
</tr>
</tbody>
</table>

The reader is assigned at the beginning of the run and released when the parameters have been read and checked.

The file is assigned when #XPLY is loaded into store and released at the end of the run. However, #XPLY can be loaded from the P.L.T. (MT0) in which case the overlays are copied on to this file and are held there until either ED1 or ED2 is full or until the end of the run.

The file is opened if, when object program output is being produced, the compiler encounters a program whose steering segment contains a #OVERLAY directive but no #ORDER directive.

The file is opened when an OUT parameter is used and released at the end of the run.

The line printer is assigned at the beginning of the run and released at the end of the run.

The magnetic tape is assigned at the beginning of the run and released when the compiler is deleted.

The magnetic tape is assigned when an OUT parameter is used and released at the end of the run.
NAME
#XPLZ

HARDWARE REQUIREMENTS
13248 words of core store
1 paper tape reader or 1 card reader
1 line printer
1 E.D.S. or Twin E.D.S.
1 E.D.S., Twin E.D.S. or F.D.S.
1 E.D.S., Twin E.D.S. or F.D.S. (optional)

DESCRIPTION
PLAN 3, batch, overlay
16K minimum processor size

Input
One or more source and/or semi-compiled programs on E.D.S., Twin E.D.S. or F.D.S. written by a DISC COSY editor (#XPMY or #XPMZ) or a sort/merge generator (#XSEG) or a user program with the file conforming to the specification for DISC COSY.

Output
OBJECT PROGRAM Unconsolidated semi-compiled object on E.D.S., Twin E.D.S. or F.D.S. in subfile format.
A binary program can be obtained by a separated run using #XPCK
LISTING On line printer

CONTROL
Entry points
20 for paper tape
21 for cards

Switch settings
18 if LIST or SHORTLIST and throw to new page required when channel 8 of control loop detected
21 if LIST or SHORTLIST and listing required on 96 print position printer

Input parameters
Mandatory IN, INE, INF, PEND
Optional OUT, OUTE, OUTF, APP, APPE, APPF, REN, WORK, SUB, NEXT, BIN, BINE, BINF, LIB, LIBE, LIBF

Steering line facilities
LIST, SHORTLIST, OBJECT, MONITOR, MAP, FULLLIST
**USE OF PERIPHERALS**

<table>
<thead>
<tr>
<th>Device</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR0 or TR0</td>
<td>parameters</td>
</tr>
<tr>
<td>LP0</td>
<td>listing and error messages</td>
</tr>
<tr>
<td>ED0</td>
<td>overlays</td>
</tr>
<tr>
<td>ED2 or FD2</td>
<td>object output (optional)</td>
</tr>
<tr>
<td>ED4 or FD4</td>
<td>source and/or semi-compiled input</td>
</tr>
</tbody>
</table>

The reader is assigned at the beginning of the run and released when the parameters have been read.

The line printer is assigned at the beginning of the run and released at the end of the run.

The file is assigned when #XPLZ is loaded into store and released when the compiler is deleted.

The file is opened when an OUT, OUTE, OUTF, APP, APPE or APPF parameter is used and released at the end of the run.

The file is assigned at the beginning of the run and released at the end of the run.
Should any additions or amendments to this Manual be necessary, they will be issued in the form of Amendment Lists. When such changes have been incorporated, an appropriate entry should be made in the table below and the pink instruction sheet(s) should be filed behind this page.

<table>
<thead>
<tr>
<th>List number</th>
<th>Item number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>to</td>
</tr>
</tbody>
</table>