Chapter 13 System Macros

This chapter describes those system macros that make available to the user standard software programs, that is, programs that can be run outside GEORGE. Macros that use programs peculiar to the GEORGE environment are described in Chapter 12.

A description of recommended standards for macro writing is also included. The purpose of these standards is to reduce the diversity in macro parameters, and to help users to write their own system macros and to understand macros issued by ICL. (Note that these are recommendations only and ICL macros will not always conform to them; for individual macros users should consult the appropriate specifications.)

A general description of the use and format of system macros is given, followed by detailed treatment of some individual ICL macros in alpnabetical order. In general system macro specifications are given in the ICL 1900 Series manual that describes the software run by the macro.

Note: The LINGO compiler macro, used in system macro examples, is fictitious. The use of actual macros such as FORTRAN was avoided to prevent inaccuracies which might arise from future changes to the specifications of system macros.

THE FORMAT OF A SYSTEM MACRO COMMAND

The format of a system macro command is basically the same as that of a built-in command; that is, a *verb* optionally preceded by a *label* and normally followed by one or more *parameters*. Since, however, the verb of a system macro command is the name of a file, the verb cannot be abbreviated to two characters.

The name of each verb indicates the function of the verb; for example, the JEAN verb is used to run one of the JEAN programs.

Conventions for specifying parameters for system macros

Certain conventions are used for specifying the parameters of most system macros. These are described below. Parameters for those system macros that do not follow these conventions are fully described in the specifications of these commands later in the chapter.

PARAMETER IDENTIFICATION

Each parameter begins with a group of characters that serve to identify the parameter.

Spaces are ignored both within the character group and preceding or following it; for example, if the characters *TR identify the file description G3PROG7 in a parameter of a system macro, the complete parameter may be written in any of the following ways:

```
*TRG3PROG7
*TR  G3PROG7
     *TR          G3PROG7
*T    RG3PROG7
```

The reason for adopting these conventions is that it enables the user, subject to certain rules, to specify parameters in any order. Since a macro may have up to 24 parameters, it would be difficult for the user to remember any fixed order for parameters. If an optional parameter is omitted, no null parameter is necessary in its place.

PARAMETERS REFERRING TO FILES

The list which follows describes the standards applying to parameters that refer to files.

1 LEADING CHARACTERS This is either an asterisk * or a hash mark # according to whether it refers to a file to be used by the piece of software interfaced or by a generated program (compiler and compiled

program respectively, for instance). Thus the call to a compiler might look as follows:

LINGO▽*CRSOURCE,BIN,#LPRESULTS

if the macro LINGO not only ran the compiler and consolidator, but also the object program in the file SOURCE.

2 IDENTIFYING ON-LINE FILES: DOCUMENTS, MAGNETIC TAPES AND EXOFILES The document name, etc, should be enclosed in brackets if this is not already required by the nature of the file name (UDAS exofile names are not given two sets of brackets). In the case of on-line magnetic tapes, to make the presence or absence of a serial number easier to detect, there will always be a serial number field, containing either the serial number or a question mark. For example:

```
LINGO▽*CR(ATLAS),BIN(77023,BINARY),
        #TRSTEER,#MT(?,DATA1),
        #MTDATA2,#GP(RESULTS),
        #MT(444,DATA3)
```

Note that here DATA2 is an off-line file, as indicated by the absence of brackets.

In order to test whether a file is on-line or off-line, some care is needed in the use of delimiters. Since it is necessary to test for the presence of a left bracket, brackets cannot be used on the right-hand side of the test; but leading spaces are ignored (see item 2 of *Parameter identification*, above), so delimiters must be either // or ??.

Example

```
IF▽STRING(%(*DA))>/(/,ONLINE▽*DA0,%(*DA)
IF▽NOT▽STRING(%(*DA))>/(/,ASSIGN▽*DA0,%(*DA)
```

3 JOB SOURCE AND MONITORING FILE In the case of basic peripherals, no file name after an input parameter means 'read from job source'. No file name after an output parameter means 'send to monitoring file'.

4 INPUT Most programs will accept input on either cards or paper tape, and some will accept both, switching from one to the other. Most macro commands require, as the first parameter, the file description of the basic peripheral file for the initial entry to the program. If a second file is to be used and is of a different type, the file description of this file may be written as any other parameter, preceded by the identifying characters appropriate to the peripheral type of the file.

For example, suppose a user has input a LINGO source program from paper tape to a file named G3PROG7 and the program description has been input from cards to a file named G3STEER(2). If the last line of the program description is the directive:

READ FROM (TR)

the program can be compiled by means of the command:

LINGO *CR G3STEER(2),*TR G3PROG7,BIN

5 ABSENCE OF BASIC INPUT PARAMETERS Where there would be at most one input parameter for either basic input peripheral, and where it would be ONLINEd to the job as in 4, a further simplification is possible:

(a) If a file name is omitted following a *CR parameter identifier, the parameter identifier may be omitted as well.

(b) If the parameter identifier is *TR and the macro could have no parameter identifier *CR, then *TR may be omitted altogether.

Examples

```
SORT▽IN*MT INPUT, OUT*MT OUTPUT
```

where the steering lines are cards in the job source.

```
PTPRINT▽*LPLIST
```

where a reel of paper tape is spliced in after the macro call.

6 ABSENCE OF A BASIC OUTPUT PARAMETER If an output file parameter is missing, then a workfile should be CREATEd, ASSIGNed to the appropriate channel, LISTFILEd to the appropriate device and ERASEd. For example:

```
LISTDISCFILE *CR, *DA(12345,INFO)
CARDDUMP BINPROGFILE
HARDCOPYT *CRDATA
```

would produce listings on a LP, CP and TP respectively, whereas

```
LISTDISC *CR, *DA(12345,INFO), *LPLIST
```

would produce listings in off-line files to be output as required by the user.

7 IN AND OUT PREFIXES In cases where it is not clear whether a magnetic media file is input or output, IN and OUT should be prefixed to the parameter defined above. This would happen in a sort, for example:

```
SORT∇*TRSTEER, IN*DA(RAW), IN*DA (MORE) , IN*MT
        (?,MTDATA), OUT*MTSORTED
```

8 CHANNEL NUMBERS It will happen that not all the files to be used on a particular device type are to be on the same channel of that type. For instance, in compilation macros, all the source from various card files is to be read on CR0, whereas the object program produced may merge streams of card input on various channels. There are two ways of doing this:

(a) If the number of options is small, insert the channel number between the parameter identifier and the file name, if any. For example:

```
UPDATE∇*CR0BF, *CPCF, *CR1IP, *LP0,
        *LP1MAINFILELIST
```

(b) If there is a large number of options (for example, all possibilities for a given peripheral type when running an object program after compilation), enclose the channel number and the file name, separated by a comma, in parentheses:

```
LINGO∇*CRSOURCE; BIN, *TR(14,DATA), #LP(6),
        #MT  (3,(5,MDATA))
```

9 DEDICATED PARAMETERS A few commonly used facilities might have a special parameter identifier; BIN in the example above is one instance. These identifiers are subject to the modifications described in 2 and 7 above for on-line files and channel numbers. Only four are recommended as standards at the time of writing:

(a) BIN and BIN*file name*: indicates to a direct access compilation macro that a consolidation is required, and whether it is to be into core (BIN alone) or into the file specified by *file name*.

(b) COMP*file name*: indicates to a compilation macro that the semi-compiled program is to be sent to a specific file rather than to a work file which will be subsequently erased.

(c) RS*file name*: indicates to a macro that it is to load a program from the named saved file

(d) LO*file name*: indicates to a macro that it is to load a program from the named file.

Examples

```
LINGO∇*CR,COMP(23,SEMI),BINPROG
LINGO∇*TR,COMPSEMICOMP,BIN
SORT∇*CRSTEER, IN*DADATA,RSXSEGPRODUCT,OUT*DA,SORTED
RUNPROG∇LOPROGRAM∇FRED,*CRINPUT,*LPOUTPUT
```

OTHER PARAMETERS

Errors and error parameters

There are three kinds of error that may occur in a macro:

1 Errors in the parameters of the macro that are detected by special checking procedures in the macro.

2 Command errors in commands within the macro. These will arise from errors in the parameters of the macro that are not detected by special checking procedures.

3 Logic errors in the program run by the macro which cause the program to be terminated.

The initial effect of an error is to generate a message relating specifically to the error that has occurred. Command errors generate standard command error messages (as described in Chapters 11 and 12). Program errors generate the messages that are standard for the program in question (as described in the relevant program manual). In the case of parameter errors detected by special checking procedures within the macro, special messages are not always sent. It may be necessary for the user to examine the job's monitoring file after an error has occurred in order to discover the precise cause of the error.

Next, for most errors other than command errors in a MOP job (for which see below), the standard message:

ERROR IN MACRO *macro name*: MACRO ABANDONED

is sent. GEORGE then goes to the label specified at the end of the macro unless an *error label parameter* is given.

ERROR LABEL PARAMETERS

The error label parameter allows the user to specify a label to which his macro is to branch if it detects an error. The standard format is

ER*label*

Example

```
LINGO∇*CR,BIN#CR,#LP,ER1HELP
                .
                .
                .
1HELP∇DP∇0,I∇HAVE∇FAILED
```

Note: An error label specified in a macro command must not be the same as a label within the macro itself, since this would lead to an error. Therefore, internal labels should always begin with the character 9. Provided the user chooses an error label beginning with any other character, the possibility of accidental duplication is eliminated.

Although the error parameter is always optional, there is one case in which there is a special reason for not omitting it.

This is the case in which the macro is issued from the job description file and input data is read from the job source. In this case, if no error parameter has been specified and the macro call and input data are not enclosed by command delimiters when an error occurs in the macro, the command processor will attempt to obey the first line of input data after the macro command as if it were a command.

Command errors in a MOP job

Command errors in a system macro issued in a MOP job cause an automatic break-in unless a WHENEVER command is set at a higher command processor level. If a WHENEVER is set, it will be obeyed. When an automatic break-in occurs, the user may either correct the erroneous command and CONTINUE, or abandon the macro by means of a QUIT command. In the latter case, the state of the workfile stack may be different from when the macro was issued.

The LIMIT parameter

If a user is expecting a lot of output to a basic peripheral file or a magnetic tape file, and the installation parameter VOLUME is set low, he will want to ASSIGN the file with a LIMIT qualifier. Alternatively he may want to use it to safeguard himself from too much output. In the case when the user specifies a file name, one possibility is to add a LIMIT qualifier to the file name in the parameter, for example

```
LISTMT∇ (MTDATA,*LPLISTING(LIMIT5000))
```

When the ASSIGN is done, the LIMIT parameter automatically turns up in the right place.

This will not work, however, if the macro wants to use an APPEND qualifier (as when a compilation wants to add the consolidator listing to the compiler listing), or if the user omits a basic output parameter (see *Parameters referring to files*); in these cases a separate LIMIT parameter is required:

```
LINGO∇*CR,BIN,LIMIT5000
```

The CORE parameter

Certain pieces of software run differently according to the size of core into which they are loaded. The LOAD command caters for this with a second parameter giving the core size. In cases when the user might need to alter this the CORE parameter is provided:

```
SORT∇*CR,IN*MTRAW,OUT*MTSORTED,CORE2000
```

INTERNAL STANDARDS

These standards apply to the coding of macros.

1 THE WORKFILE STACK This should be left as it was when the macro was entered; that is, all macro CREATEd workfiles should be ERASEd and the WORKFILEMOVE alterations repositioned.

2 Since macros may call macros, and any one of them may be CREATEing or ERASEing workfiles, when specifying a workfile as a file name in a macro parameter the user must specify the stack depth of the file at the time of assignation, not at the macro call time. The user must therefore know what the change in depth between the call and assignation is for each file name parameter in case he wants to use a workfile.

This must be included in the specification of the macro. In order to keep this within bounds, every effort should be made to keep this displacement constant over all the possible file name type parameters for the macro and, whenever possible, to make the constant zero. These things can be achieved by the macro ASSIGNing files from parameters before CREATEing any workfiles, or after ERASEing all of its own, or by arithmetic SETPARAMs. These three devices are illustrated in the example below, showing a macro which is to print magnetic tapes and has two parameters, *MT*file name* and *LP*file name* with the usual option of omitting the latter. The use that must be catered for is

```
MACRO *MT!
```

The three ways of coding this are:

(a)
```
ASSIGN∇*MT0,%(*MT)
CREATE∇!
ASSIGN∇*LP0,!
LISTFILE∇!,*LP
ERASE∇!
ENTER
```

(b)
```
CREATE∇!
ASSIGN∇*LP0,!
LISTFILE∇!,*LP
ERASE∇!
ASSIGN∇*MT0,%(*MT)
ENTER
```

(c)
```
SETPARAM∇C, (0)
CREATE∇!
SETPARAM∇C,VALUE(%C+1)
IF∇NOT∇STRING(%(*MT))>(!),GO∇1MACRO
SETPARAM∇D,VALUE(%C+%(*MT!))
SETPARAM∇(*MT),(*MT!%D)
1MACRO  ASSIGN∇*MT0,%(*MT)
```

ASSIGN▽*LP0,!

ERASE▽!

SETPARAM▽C,VALUE(%C−1)

where in the last case %C is used to remember the displacement and %D as workspace.

3 CORE IMAGES These should be deleted at the end of the macro unless it is part of the specification of the macro that it leaves a core image, as would be the case, for instance, in compiler macros consolidating to core.

4 WHENEVER▽COMMAND▽ERROR This should not be used to cover command errors that could be corrected by the user in BREAK-IN context at a MOP console. For instance, if an ASSIGN fails because the user misspelt one of his file parameters, and no WHENEVER COMMAND ERROR is set, the job will be broken in on after the command in which the error occurred and the user can then issue the ASSIGN correctly and CONTINUE. Since the WHENEVER will still probably be needed for off-line jobs, the form

IF▽NOT▽MOP, WHENEVER COMMAND ERROR

should be used.

CONTEXTUAL RESTRICTIONS ON THE USE OF SYSTEM MACROS

The following contexts are forbidden to all the system macros described in this chapter:

BREAK-IN

NO USER

PROGRAM

Further contextual restrictions are described in the specifications of the individual system macros later in the chapter.

PROGRAM COMPILATION MACROS

Parameter types

The following parameter types are associated with the program compilation macros:

SOURCE PROGRAM

LISTING

COMP

BIN

ERROR

SEMI

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, the effect of omitting it is described; if it is 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.

SOURCE PROGRAM

Position

If there is only one Source Program 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 Source Program parameter, but additional parameters of this type may occupy any succeeding position(s) in the parameter string. There must be at least one Source Program parameter; otherwise an error will result.

Format

A Source Program 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). Thus the format is:

*CR *file description*

*TR *file description*

*CR

or

*TR

Function

A Source Program parameter defines a file containing source program (if the file description is included) or specifies that source program is to be obtained from the job source (if the file description is omitted). The source program may be obtained partly from the job source and partly from a file by including the appropriate parameters.

Notes:

Any number of parameters of either peripheral type may be specified subject to the following restrictions:

1 The first parameter of the macro must be a Source Program parameter. The compiler will be entered at the entry point appropriate to the peripheral type.

2 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.

3 All the parameters of one peripheral type must be written in the correct logical order.

4 If source program 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.

5 The total number of parameters in the macro must not exceed 24.

Example 18/1

A LINGO 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 and C5 must be:

READ FROM (TR)

and the last record of T3, T6, C1 and C4 must be:

READ FROM (CR)

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:

LINGO *CR C1, *TR T3, *TR T6, *CR C2, *CR C4, *CR C5, *CR C7, COMP FSC37

Example 18/2

The user has punched the master segment of a LINGO program on paper tape, which has been input to the file COMP3. He would like the first compilation and test to be from his MOP terminal. He can then issue the command:

time ready LINGO *CR, *TR COMP3

ready

and type in the program description segment followed by

ready READ FROM (TR)

The master segment will then be compiled from COMP3.

LISTING

Position

The Listing parameter is optional and has no fixed position. If this parameter is omitted, the macro will use and erase a workfile.

Format

A Listing parameter consists of the characters *LP optionally followed by a file description. Thus the format is:

*LP *file description*

or

*LP

Function

A Listing parameter defines the file to which the compilation 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.

When line printer output is sent to the monitoring file system, the Paper Feed Control Characters are ignored. Furthermore any lines longer than 120 characters (excluding the PFCC) will be split, the excess characters being printed on the next line. It is not recommended that large amounts of output be sent to the monitoring file system.

Note

If more than 50 pages (245K) are to be output, a multifile must be specified in the file description; a work file cannot be used.

COMP

Position

The Comp parameter is optional and has no fixed position. If this parameter is omitted, the macro will use and erase a workfile.

Format

A Semicompiled parameter consists of the characters COMP followed by a file description. Thus the format is:

COMP *file description*

Function

A Semicompiled parameter defines a UDAS file to which semicompiled program is to be written. The file must previously have been CREATEd specifically for the macro. The usual rules with regard to overwriting a file apply.

BINARY

Position

The Binary parameter is optional and has no fixed position. If this parameter is omitted, no binary program will be produced.

Format

A Binary parameter consists of the characters BIN optionally followed by a file description. Thus the format is:

BIN *file description*

or

BIN

Function

A Binary parameter specifies that a binary program is to be produced. It defines a UDAS file to which the binary program is to be written (if the file description is included) or specifies that a core image of the program is to be created (if the file description is omitted). If a file is specified it must previously have been CREATEd specifically for the macro with a bucket size BUCK1. The usual rules with regard to overwriting a file apply.

Note

In the case of an overlay program, a UDAS file must always be specified.

ERROR

Position

The Error parameter is optional and has no fixed position. 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.

Format

An Error parameter consists of the characters ER followed by a label. Thus 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.

Notes

1 Special action is taken if a command error occurs in a macro issued in a MOP job (see *Command errors in a MOP job*, page 404).

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.

SEMI

Position

The Semi parameter is optional and has no fixed position. If this parameter is omitted, only the standard subroutine group for the language is used.

Format

SEMI *UDAS exofilename*

or

SEMI *UDAS file name*

Function

The Semi parameter defines semi-compiled input to the consolidator (#XPCK) in addition to the semi-compiled program and the appropriate subroutine group for the language.

There may be up to eleven Semi parameters.

Magnetic tape input

If the compiler in use has facilities for reading input from magnetic tapes, a magnetic tape can be made ONLINE to the compiler by the *MT parameter.

Position

The *MT parameter is optional and has no fixed position.

(For default action when it is omitted, see below.)

Format

*MT *magnetic tape description*

Function

Makes the specified magnetic tape on-line to the computer.

Before input can be read from the specified magnetic tape, the appropriate statement must have been read by the compiler; any details supplied in such a statement will be overridden by the information obtained from the *MT parameter.

It is also possible for magnetic tape to be used without an associated *MT parameter when appropriate statements are written in the source program. The compiler will issue an unanticipated open mode PERI, which GEORGE will implement. Magnetic tapes will be identified by magnetic tape name and generation number. This information will be obtained from the source program.

Only one *MT parameter can be used. The second and subsequent parameters of this type will never be accessed. Hence if more than one magnetic tape is required, the second will always be opened by an unanticipated open mode PERI.

DISC

The Disc parameter is optional and has no fixed position. If this parameter is omitted, an exofile of the required name will be opened. However, no attempt will be made to obtain an exofile unless the required disc is on-line.

Format

A Disc parameter consists of the characters *DA followed by a file or a UDAS exofile description. Thus the format is:

*DA *file description*

or

*DA *exofile description*

Function

A Disc parameter defines a UDAS filestore file or a UDAS exofile from which source program is to be read. The file specified will be opened when the appropriate statement has been read by the compiler. Any details supplied in such a statement in the source program will be overridden by the information obtained from the Disc parameter.

Note

Only one Disc parameter may be used.

Consolidation

All the compiler macros except PLAN4 use the disc consolidator program #XPCK to consolidate the semicompiled output. If the consolidation is successful, the resulting binary program can be dealt with in two ways:

1 If the programmer specifies a file name, the binary program will be stored in the UDAS file with that name.

2 If no file name is specified, the binary program, which cannot be an overlaid program, will be left in core by the consolidator.

If the program exceeds the maximum core size allowed the consolidation will fail, and the following error message will be displayed:

YOUR PROGRAM IS TOO BIG FOR ENVIRONMENT

and the macro will be abandoned.

The job can be re-run when the value of the installation parameter COREOBJECT is suitably increased (see *INSTPARA*, page 280). If, however, the program is larger than the maximum value that can sensibly be given to COREOBJECT, the programmer must find some means of reducing the core size of his program.

When a PLAN program with #PERIPHERAL directive to obtain peripherals is being compiled into core, the consolidator will attempt to implement the peripheral requests before it halts LD, and the macro will therefore display the message:

PLEASE ALLOCATE PERIPHERALS AND RESUME

In this case the job source must ASSIGN or ONLINE peripherals as required. If the command RESUME is then issued, the consolidator will move the program down to its correct place and then halt LD.

When a file is specified for binary output, #XPCK will automatically give the binary program a type 2(GO) entry block. The LOAD command does not implement such an entry block. In the case of non-overlay programs a RESUME command will merely result in the program halting with the message LD. Overlay programs, however, will always require a RESUME command following the LOAD command to allow the overlay package to be initialised. The program will then halt LD.

SPECIFICATIONS OF SYSTEM MACRO COMMANDS

The remaining pages of this chapter are devoted to the specifications of system macro commands. The commands are arranged in alphabetical order and 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 those described on page 406, 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 parameter types for use with program compilation macros. Where a parameter type has already been described under *Program compilation macros* 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 Under this heading is given a list of references to ICL 1900 Series manuals containing explanations of the error messages that may be generated in the course of the execution of the macro. If the error messages are exclusive to the GEORGE environment, they are explained in full.

BINDUMP

The specification of BINDUMP is now in Chapter 12.

CONSOLIDATE

FUNCTION

The CONSOLIDATE system macro consolidates semicompiled programs held in D.A. filestore files by using #XPCK.

FORMAT

CONSOLIDATE *parameter list*

Parameter types in the parameter list

STEERING

Position

The Steering parameter has no fixed position but must be included somewhere in the parameter list; otherwise an error will result.

Format

A Steering parameter consists of the characters AUTO or the characters *CR or *TR optionally followed by a file description of a file containing the steering information for #XPCK. Thus the format is:

AUTO

*CR

*TR

*CR *file description*

or

*TR *file description*

Function

If the parameter is AUTO, the steering information will be assumed to be set in #XPCK's accumulators. AUTO can be used only if the program event prior to the CONSOLIDATE command was caused by a 160/2 extracode (DELTY) issued by a disc compiler.

If the parameter includes a file description, the steering information will be assumed to be held in a filestore file in card or paper tape format as indicated by the preceding characters, that is *CR or *TR. If the parameter consists solely of the characters *CR or *TR, steering information will be read from the job source.

Note

Where the steering information specifies files that are also specified in the macro command, the following rules apply:

1 Files must be specified as type ED even if they were created as F.D.S. or D.A. files.

2 The file name specified in the steering information is irrelevant but a subfile name, when specified, must be correct.

BINARY

This parameter type is as described under *Program compilation macros,* page 407, with two exceptions:

1 The parameter consists of the characters BIN followed by a file or an exofile description. Thus the format is:

BIN *file description*

or

BIN *exofile description*

2 If the parameter is omitted, the object program will be loaded into core store.

PMD

Position

The PMD parameter has no fixed position and is optional.

Format

A PMD parameter consists of the characters PMD followed by a file or exofile description. Thus the format is:

PMD *file description*

or

PMD *exofile description*

Function

A PMD parameter specifies the file to which the Algol postmortem dump information is to be written by the consolidator. A PMD parameter must be present if AUTO is present and if the ALGOL source program description contained a 'PMD' statement.

INPUT

Position

The Input parameter has no fixed position but must be included somewhere in the parameter list; otherwise an error will result.

Format

An Input parameter consists of the characters IN followed by a file or exofile description. Thus the format is:

IN *file description*

or

IN *exofile description*

Function

An Input parameter specifies a D.A. filestore file or exofile containing the semicompiled program to be consolidated or the library subroutines required for consolidation.

Note

There may be more than one Input parameter specifying the source of semicompiled input or more than one specifying the source of the required library subroutines. However, the total number of Input parameters must not exceed twelve.

Input parameters must be presented in the order in which the files are to be used by the consolidator, that is, in the order in which the semicompiled and library files are specified in the steering information.

If a file is to be used more than once by #XPCK as the source of semicompiled input or library subroutines it must only be described in one Input parameter.

LISTING, ERROR — These parameter types are described under *Program compilation macros*, page 406.

EXECUTION

The free-standing consolidator #XPCK is loaded. Steering information for the program is either obtained from a filestore file or passed over into the accumulators by a compiler. The program is then entered and the semicompiled program is either written to a filestore file in binary or consolidated into core store where it halts LD.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM XPCK	EXECUTE

ERROR MESSAGES

Details of error messages may be found in the #XPCK section of *Compiling Systems* Edition 1 TP4241).

EXCEPTION CONDITIONS

1 If more than 12 IN parameters are presented the following message will be sent to the monitoring file:

DISPLAY – TOO MANY "IN" PARAMETERS

Further IN parameters are ignored and the macro continues.

Consolidation will be attempted using the first 12 IN files specified.

2 If the file description for an IN parameter is omitted the following message will be sent to the monitoring file:

DISPLAY – NO FILENAME IN "IN" PARAMETER

The IN parameter is ignored and the macro continues.

3 If the file description for a PMD parameter is omitted the following message will be sent to the monitoring file:

DISPLAY – NO FILENAME IN "PMD" PARAMETER

The macro continues and a workfile is created to hold the postmortem dump information: this workfile is erased at the end of consolidation.

4 If no IN parameter with a file description is presented the following message will be sent to the monitoring file:

DISPLAY – NO VALID "IN" PARAMETER

The macro is abandoned.

5 If none of the parameters AUTO, *CR, *TR is presented the following message is sent to the monitoring file:

DISPLAY – STEERING PARAMETER IS OMITTED

The macro is abandoned.

When the macro is abandoned, following 4 or 5 above, the following message is also produced:

DISPLAY – ERROR IN MACRO CONSOLIDATE:ABANDONED.

DISCPROG

FUNCTION

The DISCPROG system macro writes overlaid binary programs to a disc file or updates an exofile using the library maintenance program #XPEU.

FORMAT

DISCPROG *parameter list*

Parameter types in the parameter list

PARAMETER INPUT

Position

The Parameter Input parameter must be the first entry in the parameter list; otherwise an error will result.

Format

A Parameter Input parameter consists of the characters *CR or *TR optionally followed by a file description. Thus the format is:

*CR

*TR

*CR *file description*

or

*TR *file description*

Function

A Parameter Input parameter defines the source of the parameter input to #XPEU as a card or paper tape file (if the file description is included) or the job source (if the file description is omitted).

FILE DEFINITION

Position

The File Definition parameter is optional and has no fixed position. If it is omitted, GEORGE assumes that an exofile is to be written to or updated. Unanticipated opening operations are used to open the exofile provided that it is on-line. If no exofile is specified in the parameter input to #XPEU, or an exofile is specified but it is not on-line, an error is generated.

Format

A File Definition parameter consists of the characters EDLIB followed by a file or an exofile description or FDLIB followed by a file description. Thus the format is:

EDLIB *file description*

EDLIB *exofile description*

or

FDLIB *file description*

Function

A File Definition parameter defines an E.D.S. or F.D.S. filestore file or E.D.S. exofile to which binary programs are to be written. The file must have been CREATEd specifically for this run of the program with a bucket size BUCK1.

In order to use the FIND command on an exofile, the exofile name must be of the form PROGRAM∇*nnnn* where *nnnn* is an expression consisting of four alphanumeric characters, the first of which must be alphabetic.

INSERTION DEFINITION

Position

The Insertion Definition parameter has no fixed position and is optional. If it is omitted, GEORGE assumes that programs are to be transcribed from magnetic tape or exofiles. GEORGE will use unanticipated opening operations to open tapes or exofiles. If an exofile is to be used, it must be on-line. If no magnetic tape or exofile is specified in the parameter input to #XPEU, or an exofile is specified but it is not on-line, an error will be generated.

Format

An Insertion Definition parameter consists of the characters *CR, *TR or *FD followed by a file description, the characters *ED followed by a file or an exofile description or the characters *MT followed by a magnetic tape description. Thus the format is:

*CR *file description*

*TR *file description*

*FD *file description*

*ED *file description*

*ED *exofile description*

or

*MT *magnetic tape description*

Function

An Insertion Definition parameter defines a file or magnetic tape or E.D.S. exofile from which a program is to be transcribed. If an exofile is to be written to, more than one *CR or *TR parameter may be given (see *Note*).

Note

If the programs are to be transcribed to an exofile, more than one Insertion Definition parameter may be given. In this case all the Insertion Definition parameters of one particular type must be given in the order in which the program will open the files.

LISTING, ERROR — These parameter types are as described under *Program compilation macros*, page 406, except that the listing is not a compilation listing.

EXECUTION

The library maintenance program #XPEU is loaded and entered. Binary programs are transcribed from one of several input media to the specified disc file or exofile.

Note: The use of the DISCPROG macro to write to filestore files is restricted in the current mark in two ways:

1 The LOAD command can load only the first program in a disc file. Hence only one program may be written to a disc file by means of DISCPROG, if it is subsequently to be LOADed.

2 #XPEU updates a file by zeroizing the name of the old program and adding the new one. Hence it is not possible to update a program file.

These restrictions do not apply to exofiles.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM▽XPEU	EXECUTE

ERROR MESSAGES

Details of error messages may be found in the #XPEU section of *Compiling Systems* (Edition 1 TP4241).

DISCSUB

FUNCTION

The DISCSUB system macro writes library subroutines to a disc file or exofile, or updates the file or exofile using the library maintenance program #XPES.

FORMAT

DISCSUB *parameter list*

Parameter types in the parameter list

PARAMETER INPUT

Position

The Parameter Input parameter must be the first entry in the parameter list; otherwise an error will result.

Format

A Parameter Input parameter consists of the characters *CR or *TR optionally followed by a file description. Thus the format is:

*CR

*TR

*CR *file description*

or

*TR *file description*

Function

A Parameter Input parameter defines the source of the parameters input to #XPES as a card or paper tape file (if the file description is included) or the job source (if the file description is omitted).

FILE DEFINITION

Position

The File Definition parameter is optional and has no fixed position. If it is omitted, GEORGE assumes that an exofile is to be written to or updated. Unanticipated opening operations are used to open the exofile, provided that it is on-line. If no exofile is specified in the parameter input to #XPES, or an exofile is specified but it is not on-line, an error is generated.

Format

A File Definition parameter consists of the characters EDLIB followed by a file or an exofile description or FDLIB followed by a file description. Thus the format is:

EDLIB *file description*

EDLIB *exofile description*

or

FDLIB *file description*

Function

A File Definition parameter defines an E.D.S. or F.D.S. filestore file or an E.D.S. exofile which is to be written to or updated.

INSERTION DEFINITION

Position

The Insertion Definition parameter is optional and has no fixed position (see *Note*). If it is omitted, GEORGE assumes that subroutines are to be transcribed from magnetic tape or an exofile and GEORGE will use unanticipated opening operations to open a tape or exofile. If an exofile is to be used, it must be on-line. If no magnetic tape or exofile is specified in the parameter input to #XPES or an exofile is specified but it is not on-line, an error will be generated.

Format

An Insertion Definition parameter consists of the characters *CR, *TR or *FD followed by a file description, *ED followed by a file or an exofile description or the characters *MT followed by a magnetic tape description. Thus the format is:

*CR *file description*

*TR *file description*

*FD *file description*

*ED *file description*

*ED *exofile description*

or

*MT *magnetic tape description*

Function

An Insertion Definition parameter defines a file or magnetic tape from which a subroutine or group of subroutines is to be transcribed. More than one *CR or *TR parameter may be given (see *Note*).

Note

All the Insertion Definition parameters of one particular type must be given in the order in which the program will open the files

LISTING
ERROR

} These parameter types are as described under *Program compilation macros*, page 406, except that the listing is not a compilation listing.

EXECUTION

The library maintenance program #XPES is loaded and entered. Subroutines are transcribed from one of several input media to the specified disc file or exofile.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM▽XPES	EXECUTE

ERROR MESSAGES

Details of error messages may be found in the #XPES section of *Compiling Systems* (Edition 1 TP4241).

DUMPROG

FUNCTION

The DUMPROG system macro produces in a disc exofile, on magnetic tape, on cards or on paper tape a selective copy of a disc program file or exofile in a disc, card or paper tape file, using the library maintenance program #XPEC.

FORMAT

DUMPROG *parameter list*

Parameter types in the parameter list

PARAMETER INPUT

This parameter type is as described under DISCPROG except that the library program involved is #XPEC instead of #XPEU.

LIBRARY FILE

Position

The File Definition parameter is optional and has no fixed position. If it is omitted, GEORGE assumes that an exofile is to be used. Unanticipated opening operations are used to open the exofile, provided that it is on-line. If no exofile is specified in the parameter input to #XPEC, or if an exofile is specified but it is not on-line, an error will be generated.

Format

A File Definition parameter consists of the characters EDLIB followed by a file or an exofile description or FDLIB followed by a file description. Thus the format is:

EDLIB *file description*

EDLIB *exofile description*

or

FDLIB *file description*

Function

A File Definition parameter defines an E.D.S. or F.D.S. filestore file or E.D.S. exofile from which programs are to be copied.

OUTPUT

Position

The Output parameter is optional and has no fixed position. If it is omitted, GEORGE assumes that the copy is to be written to an exofile. Unanticipated opening operations are used to open the exofile, provided that it is on-line. If no exofile is specified in the parameter input to #XPEC, or if an exofile is specified but it is not on-line, an error will be generated. (Note that magnetic tape may be used only if the parameter has the form

*MT *magnetic tape description*

see *Format*.)

Format

An Output parameter consists of the characters *CP or *TP optionally followed by a file description, the characters *ED followed by a file or an exofile description, the characters *FD followed by a file description or the characters *MT followed by a magnetic tape description. Thus the format is:

*CP

*TP

*CP *file description*

*TP *file description*

*ED *file description*

*ED *exofile description*

*FD *file description*

or

*MT *magnetic tape description*

Function

An Output parameter defines an on-line card or paper tape punch (if the characters *CP or *TP appear alone as the parameter) to which the copy is to be output or defines a file, exofile or owned magnetic tape to which the copy is to be written. Any number of output parameters may be given, but not more than one of each peripheral type.

LISTING
ERROR

These parameter types are as described under *Program compilation macros*, page 406, except that the listing is not a compilation listing.

EXECUTION

The library maintenance program #XPEC is loaded and entered. Programs are selectively copied from the disc file or exofile in accordance with the parameters specified.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM▽XPEC	EXECUTE

ERROR MESSAGES

Details of error messages may be found in the #XPEC section of *Compiling Systems* (Edition 1 TP4241).

MTEDIT

Function

The MTEDIT system macro edits magnetic tapes.

Format

MTEDIT *parameter list*

Parameter types in the parameter list

EDITING FILE

Position

The Editing File parameter must be the first entry in the parameter list. If it is omitted, an error will result.

Format

An Editing File parameter consists of the characters *CR or *TR optionally followed by a file description. Thus the format is:

*CR *file description*

*TR *file description*

*CR

or

*TR

Function

An Editing File parameter specifies that the editing file for #XKYA is held in the file identified by the file description or that the editing file will be read from the job source (if the file description is omitted).

OLD MASTER TAPE

Position

The Old Master Tape parameter is optional and has no fixed position. If it is omitted, the requisite magnetic tape will be opened by an unanticipated open mode PERI, that is, by an OFR or OFW line in the editing file.

Format

An Old Master Tape parameter consists of the characters *MT0 followed by a magnetic tape description. Thus the format is:

*MT0 *magnetic tape description*

Function

An Old Master Tape parameter specifies that the old master tape is that identified by the magnetic tape description.

OUTPUT TAPE

Position

The Output Tape parameter is optional and has no fixed position. If it is omitted, the requisite tape will be opened by an unanticipated open mode PERI, that is, by an OFW line in the editing file.

Format

An Output Tape parameter consists of the characters *MT1 followed by a magnetic tape description. Thus the format is:

*MT1 *magnetic tape description*

Function

An Output Tape parameter specifies that the output tape is that specified by the magnetic tape description.

LISTING
ERROR

These parameter types are as described under *Program compilation macros*, page 406.

EXECUTION

The magnetic tape editor program #XKYA is loaded and any magnetic tapes specified in the parameter list are ONLINEd. The program is then entered. Other magnetic tapes are opened by unanticipated open mode PERIs.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM▽XKYA	EXECUTE

ERROR MESSAGES

Details of error messages may be found in the ICL 1900 Series manual, *Compiling Systems* (Edition 1, TP4241).

MTLIB

FUNCTION

The MTLIB system macro updates program library tapes in subfile format.

FORMAT

MTLIB *parameter list*

Parameter types in the parameter list

PARAMETER INPUT

Position

The Parameter Input parameter must be the first entry in the parameter list; otherwise an error will result.

Format

A Parameter Input parameter consists of the characters *CR or *TR optionally followed by a file description. Thus the format is:

*CR *file description*

or

*TR *file description*

or

*CR

or

*TR

Function

A Parameter Input parameter specifies that the parameter input to #XFMV is held in the file identified by the file description or that it is to be read from the job source (if the file description is omitted).

INSERTION DEFINITION

Position

The Insertion Definition parameter has no fixed position but must be included if insertions from cards or paper tape are to be used.

Format

An Insertion Definition parameter consists of the characters *CR or *TR followed by a file description. Thus the format is:

*CR *file description*

or

*TR *file description*

Function

An Insertion Definition parameter defines, by the file description, a card or paper tape file from which programs or subroutines are to be inserted.

Notes

1 There must be one parameter for each card or paper tape file from which programs or subroutines are to be inserted.

2 There may be any number of each type of parameter; however, parameters of any one type must be in the order in which the files are required because the macro looks for the next parameter of the required type.

MAGNETIC TAPE

Position

The Magnetic Tape parameters have no fixed position and are optional. If omitted, information about magnetic tapes will be taken from #XPMV parameters.

Format

A Magnetic Tape parameter consists of the characters OLMT, OUMT or RDMT followed by a magnetic tape description. Thus the format is:

OLMT *magnetic tape description*

or

OUMT *magnetic tape description*

or

RDMT *magnetic tape description*

Function

A Magnetic Tape parameter causes the specified magnetic tape to be made ONLINE to the program. If the characters OLMT are specified, the old tape to be updated is on-lined; if the characters OUMT are specified, the new tape is on-lined; if the characters RDMT are specified, an insertion tape is on-lined.

Notes

1 Any of the types of Magnetic Tape parameters may be used, either singly or together with any of the others.

2 The advantages of using the Magnetic Tape parameters is that they allow specific tapes to be used.

LISTING

Position

The Listing parameter is optional and has no fixed position. If this parameter is omitted, the macro will use and erase a workfile.

Format

A Listing parameter consists of the characters *LP optionally followed by a file description. Thus the format is:

*LP *file description*

or

*LP

Function

A Listing parameter defines the file to which the compilation 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 over-writing it.

When line printer output is sent to the monitoring file system, the Paper Feed Control Characters are ignored. Furthermore any lines longer than 120 characters (excluding the PFCC) will be split, the excess characters being printed on the next line. It is not recommended that large amounts of output be sent to the monitoring file system.

ERROR

Position

The Error parameter is optional and has no fixed position. 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.

Format

An Error parameter consists of the characters ER followed by a label. Thus 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.

Notes

1 If a command error occurs when the macro has been issued in a MOP job, an automatic break-in will occur unless a WHENEVER command is set at a higher command processor level. If a WHENEVER is set, it will be obeyed. When an automatic break-in occurs, the user may either correct the erroneous command and CONTINUE, or abandon the macro by means of a QUIT command.

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.

EXECUTION

The library maintenance program #XPMV is loaded and entered. Magnetic tapes are opened either by specifying the tapes required in the macro parameters or by #XPMV generating unanticipated opening operations according to the #XPMV parameters.

LIBRARY ENTRANTS ACCESSED

Entrant	Access modes
:LIB.PROGRAM▽XPMV	EXECUTE

ERROR MESSAGES

Details of error messages can be found in the specification for #XPMV in the manual *Compiling Systems* (Edition 1 TP4241).

FUNCTION

The RUN system macro is used to load and run a program or run the current core image. It enables the user to run a simple program with card or paper tape input and line printer output by issuing one macro instead of the commands LOAD, ASSIGN, ENTER etc.

FORMAT

RUN *parameter list*

Parameter types in the parameter list

BINARY PROGRAM

Position

The Binary Program parameter has no fixed position and is optional. If it is omitted, GEORGE assumes that the current core image is to be run.

Format

A Binary Program parameter consists of the character # followed by a file description. Thus the format is:

#file description

Function

A Binary Program parameter defines a file from which a binary program is to be loaded.

CARD READER

Position

The Card Reader parameter is optional and has no fixed position. If both this and the Tape Reader parameter are omitted, the job source will be ONLINEd to the program as *CR0.

Format

A Card Reader parameter consists of the characters *CR optionally followed by a file description. Thus the format is:

*CR

or

*CR *file description*

Function

A Card Reader parameter defines a file to be ASSIGNed to the program as *CR0 (if the file description is included) or specifies that the job source is to be ONLINEd to the program as *CR0 (if the file description is omitted).

TAPE READER

Position

The Tape Reader parameter is optional and has no fixed position. If both this and the Card Reader parameter are omitted, the job source will be ONLINEd to the program as *CR0.

Format

A Tape Reader parameter consists of the characters *TR optionally followed by a file description. Thus the format is:

*TR

or

*TR *file description*

Function

A Tape Reader parameter defines a file to be ASSIGNed to the program as *TR0 (if the file description is included) or specifies that the job source is to be ONLINEd to the program as *TR0.

LINE PRINTER

Position

The Line Printer parameter has no fixed position and is optional. If it is omitted, the job's monitoring file will be ONLINEd to the program as *LP0.

Format

A Line Printer parameter consists of the characters *LP optionally followed by a file description. Thus the format is:

*LP

or

*LP *file description*

Function

A Line Printer parameter defines a file to be ASSIGNed to the program as *LP0 (if the file description is included) or specifies that the job's monitoring file is to be ONLINEd to the program as *LP0 (if the file description is omitted).

ENTRY

Position

The Entry parameter is optional and has no fixed position. If it is omitted, entry point 0 (Word 20) is assumed.

Format

An Entry parameter consists of the characters ENTRY followed by a number between 0 and 9. Thus the format is:

ENTRY *number*

Function

An Entry parameter specifies the entry point for the program run.

TIME

Position

The Time parameter is optional and has no fixed position. If it is omitted, the program will be allowed five seconds of mill time.

Format

A Time parameter consists of the characters TIME followed by an amount of mill time. Thus the format is:

TIME *milltime*

Function

A Time parameter sets a mill time limit on the program run.

END

Position

The End parameter is optional and has no fixed position. If it is omitted, any HALTED or DELETED message will cause control to pass to the command after RUN.

Format

An End parameter consists of the characters END followed by text. Thus the format is:

END *text*

Function

The text consists of an expected HALTED or DELETED message in the program. If the program ends in the expected way, control will pass to the command after RUN in the job description. The action taken in the event of an unexpected message or failure depends on whether or not an Error parameter (see below) has been included. The full text of the expected message need not be given in this parameter; it is sufficient that the first *n* characters of the message should coincide with the text, where *n* is the number of characters in the text. Spaces are significant.

ERROR

Position

The Error parameter has no fixed position and is optional. If it is omitted, the command after RUN will be obeyed, if the run does not end as expected or if an error is detected in the macro.

Format

An Error parameter consists of the characters ER followed by a label. Thus the format is:

ER *label*

Function

An Error parameter defines a label to which control is to be passed if:

1 The program halts or is deleted and the END message comparison fails (see above).

2 There is a failure during the program.

3 There is an error in the macro.

LIBRARY ENTRANTS ACCESSED

None.

ERROR MESSAGES

The error messages are as programmed plus the run time error messages incorporated by the compiler, if any.

SOURCEDIT

FUNCTION

The SOURCEDIT system macro is used to edit source programs by line number.

FORMAT

SOURCEDIT *parameter list*

Parameter types in the parameter list

OLD FILE

Position

The Old File parameter must be the first entry in the parameter list; otherwise an error will result.

Format

An Old File parameter consists of the characters *CR or *TR followed by a file description. Thus the format is:

*CR *file description*

or

*TR *file description*

Function

An Old File parameter defines the file containing source program to be edited. If the type is *TR, the file is assumed to be in full mode. The file must be terminated by a line beginning with four asterisks.

NEW FILE/DOCUMENT

Position

The New File/Document parameter must be the second parameter; otherwise an error will result.

Format

A New File/Document parameter consists of a file description or a document name. Thus the format is:

file description

or

document name

Function

If the Old File parameter is a *CR parameter, a New File/Document parameter defines the file to contain the edited source program. This file will be created by the macro if it does not already exist.

If the Old File parameter is a *TR parameter, it defines the paper tape document that will be output containing the edited source program. This is necessary in the current mark because TP files cannot be read as TR files.

AMENDMENTS

Position

The AMENDMENTS parameter is optional and has no fixed position. If it is omitted, amendment lines will be read from the job source and assumed to be in the peripheral type and mode of the old file to be edited.

Format

An Amendments parameter consists of the characters AMENDMENTS followed by a file description. Thus the format is:

AMENDMENTS *file description*

Function

An AMENDMENTS parameter defines the file containing the amendment lines. The peripheral type and mode of the file must be the same as the peripheral type and mode of the old file that is to be edited. (See *Execution*, below, for details of the amendments.)

LISTING

This parameter type is as described under *Program compilation macros*, page 406, except that the listing is not a compilation listing.

EXECUTION

The editing program #XMEG is loaded and the required peripheral connections are made. The program is then entered at the appropriate entry point (word 20 for paper tape, word 21 for cards).

Amendments

Amendments consist of *ALTER instructions followed by any source lines to be inserted. The amendments are terminated by a line beginning with four asterisks.

The *ALTER instruction indicates that source lines are to be deleted, inserted or amended. Lines are referenced by line numbers, the first line of the old file being line one.

The *ALTER instruction may take the following forms:

1 Where lines are to be deleted:

*ALTER *l-m*

or

*ALTER *l,n*

l = the line number of the first line to be deleted.

m = the line number of the last line to be deleted.

n = the number of lines to be deleted.

The directive will be followed by any lines that are to replace the deleted lines. All lines following the *ALTER instruction will be inserted up to the next *ALTER instruction or the terminator.

2 Where lines are to be inserted without any deletions being made:

*ALTER *l*

l = the line number of the line before which new lines are to be inserted.

If *l* is one greater than the line number of the last line of the program, the new lines will be inserted at the end of the program.

The instruction will, as before, be followed by the new lines to be inserted.

*ALTER instructions should appear in ascending order of line number and the line number of a line that has been deleted should not be used. Spaces must not precede *ALTER or ****. At least one space must be given between *ALTER and its first parameter. All numbers must be expressed as decimal integers.

OUTPUT

1 LISTING Each run of #XMEG produces a listing of the amended file with line numbers included, to assist future editing. These line numbers are not written to the new file/document. Insertion lines are denoted on the listing by asterisks. If an output record is too long to be listed in full, the character > is given before the line and the extra data is not listed. This data will, however, be written to the new file/document. Lower case letters are overprinted with apostrophes.

Where lines are deleted, the message

n LINES DELETED

is given in the listing.

2 CARD FILE OR PAPER TAPE DOCUMENT Output is terminated by a **** record. With cards, sequence numbers are copied across unchecked.

LIBRARY ENTRANTS ACCESSED

Entrant	*Modes of access*
:LIB.PROGRAM▽XMEG	EXECUTE

ERROR MESSAGES

If the run is successfully completed, the message HALTED:-OK is sent. Otherwise the following message may be given:

HALTED:-ZZ The run has been abandoned by the program. The reason is given in the listing.