EDIT (ED)

Function

Calls in the editor to edit a named basic file. The file cannot be a multifile.

Format

EDIT *oldfile, newfile, editfile*

oldfile	is the file description of the old file to be edited. This parameter is mandatory and must be the first parameter. Relevant qualifiers are NOWAIT, REPLY, FROZEN.
newfile	is the file description of the file to which the edited information is to be written. If this parameter is null, the editor will create a file with the same file description as the file described in the first parameter, except that the file generation number will be incremented by one. Relevant qualifiers are LIMIT, NOWAIT, REPLY, FROZEN, APPEND, TRAPGO, TRAPSTOP, OWNERACC.
editfile	is the file description of the source of the editing instructions. If this parameter is null, the job source is assumed as the source of the editing instructions. Relevant qualifiers are NOWAIT, REPLY, FROZEN.

Forbidden contexts

NO USER

Execution

Full details of the execution of the edit command, and of the use of the editor are given in Chapter 6.

Error messages

See Chapter 6.

ENDJOB (END or EJ)

Function

Terminates the execution of a background job.

Format

ENDJOB *action on monitoring file,* RETAIN (*local name*)

The parameters are optional. For the action on monitoring file parameters, see Tables 1 and 2, page 178.3. If RETAIN (*local name*) is given, the whole monitoring file will be copied to a file with the specified name, as by a COPY command. RETAIN may be abbreviated to RT.

Forbidden contexts

NO USER, MOP, PROGRAM

The context is changed to NO USER.

Execution

The command removes the job in which it occurs from the system's list of active jobs, closes the job description file (and also erases it if the job was initiated by a JOB command), deletes any core image and closes the monitoring file. Any temporary files are erased and temporary magnetic tapes are returned to the pool. The following message is output in the COMMENT category:

MAXIMUM ONLINE BS USED = *n* KWORDS

For the significance of the above message see *Improving the use of on-line backing store,* page 68.3. The monitoring file is then closed. At this point (providing built-in log analysis has not been specified) the log analysis program is run on the monitoring file to produce the charge for running the job.

The *action on monitoring file* parameter(s) indicates how much listing of the monitoring file is required. If none is given, everything is output.

If no ENDJOB command is met in a job and the end of the job description file is reached, the job is suspended for a period determined by the value of the WAITTIME installation parameter (see *INSTPARA)* (as if WAIT had been issued in the job). The suspension will be terminated if the job is connected to a MOP terminal by a CONNECT command before the period is up. If the job is not connected before the end of the suspension period, the job will be terminated automatically by an ENDJOB command and the whole of the monitoring file will be output.

Note: ENDJOB can be present anywhere in the job description file any number of times.

Examples

ENDJOB	The whole of the job's monitoring file will be listed.
ENDJOB NONE	None of the job's monitoring file will be listed.
END OBJECT,RETAIN (MONFILE)	Object program output in the monitoring file will be listed and the whole monitoring file will be copied to a file named MONFILE.

Error messages

z IS AN UNKNOWN MONITORING CATEGORY

z IS NOT A CORRECTLY FORMED NAME (local name)

Logging messages

GEORGE 3 only:	*time milltime* DELETED, CLOCKED *progtime*
GEORGE 4 only:	*time milltime* DELETED, CLOCKED *progtime* MAXQ = *number*K, PT = *number,* USED *number* K

CLOCKED *progtime* is output only if *progtime* is non-zero.

These messages are sent if there was a core image when the ENDJOB command was issued.

time milltime REALTIME FINISHED

This message is sent if there was a core image in real time state when the ENDJOB command was issued.

time milltime FINISHED : *number*$_1$ LISTFILES HERE, *number*$_2$ ELSEWHERE AND FURTHER LISTFILES FOR *number*$_3$ MULTIFILES

Relevant parts of the message will be omitted if *number*$_2$ or *number*$_3$ is zero.

This is the final message sent to the job's monitoring file.

ENTER (EN)

Function

Causes the current object program to be entered at the specified entry point.

Format

ENTER *number*, PARAM ($a_1, a_2, \ldots a_n$)

The *number* parameter indicates the entry point and must be between 0 and 9. The program is entered at word 20 + *number.*

$a_1, a_2, \ldots a_n$ are optional parameters to be passed to the object program when it asks for them. These must conform to the rules that govern the format of all parameters (see page 160). If the PARAM parameter is omitted, *number* may also be omitted. In this case it will be taken as zero.

Forbidden contexts

NO CORE IMAGE, PROGRAM, BREAK-IN

Execution

If loading of the current core image is not completed, the ENTER command causes the completion of the load (see *Loading and entering programs,* page 20).

Word 8 of the program is set to contain the value 20 + *number,* all other bits of Word 8 being cleared.

Word 9 of the program is cleared.

The low level scheduler is informed that the program is ready to run.

By means of special PERI instructions (see page 42), a parameter specified in the ENTER command may be requested within the object program.

Examples

ENTER 9	The program will be entered at word 29.
ENTER 0, PARAM(64,(1,2,3))	The program will be entered at word 20; 64 and (1,2,3) are object program parameters.
ENTER	The program will be entered at word 20.

Error messages

THE ENTRY POINT IS NOT BETWEEN 0 AND 9.

z HAS A CHECKSUM ERROR

z IS NOT IN BINARY FORMAT

INAPPROPRIATE/INVALID BLOCK TYPE

z HAS NO ENTRY BLOCK

z HAS A BLOCK COUNT ERROR

ENTRUST (ET)

Function

Assigns trusted status to an object program.

Format

    ENTRUST   *status*

status is that to be given, or added, to the object program: it must be Q.

Forbidden contexts

NO CORE IMAGE

Execution

A check is made to ensure that the proper user has the TRUSTED privilege. If he does, the core image will be treated as though the given status had been set in the request slip until the core image is deleted or a CANCEL ENTRUST command issued for that status (see below).

The status assigned by this command may be preserved by a SAVE command. However, ENTRUST will not alter the request slip in the file from which the program was loaded.

Error messages

TRUSTED STATUS PARAMETER MISSING

z NOT RECOGNISED

CANCELLATION OF ENTRUST

Function

Removes trusted status from an object program.

Format

    CANCEL ENTRUST *status*

Forbidden contexts

NO CORE IMAGE

Execution

After this command has been issued, the program will not have the specified status. This does not depend on whether the program actually had this status, or on how the status was assigned.

ERASE (ER)

Function

Removes a file from the filestore.

Format

ERASE *file description*

ERASE *file description*$_1$, *file description*$_2$, . . . *file description*$_n$

Each parameter must be the name of a filestore file for which the user has an ERASE trap.

Forbidden contexts

NO USER

Execution

Checks are made that

1 The user has an ERASE trap for each file specified.

2 The user is the owner of any directory specified or is a suitably privileged user acting as owner (see *Ownership of entrants,* page 175)

3 Each file specified is not user-frozen (see *User-freezing of entrants,* page 176)

After this command has been implemented no trace of the named files will be found anywhere in the system. Any directory entry will be removed and all space occupied by an ERASEd file will be freed. An open file will not be erased until it has been closed by the activity that is currently using it. After an ERASE command has been issued an attempt to set up a file with the same name as one that is to be erased will result in a wait for the erasure to take place. An attempt to use a file that is in the process of being erased will result in the message 'ENTRANT *z* DOES NOT EXIST'.

If the file description starts with '!' the workfile stack is pulled up and the workfile is erased.

If the workfile stack has not been pushed down to the depth indicated, an error is reported.

If the named file is a directory, the directory and all the files inferior to it are erased, and all magnetic tapes inferior to it are RETURNed, provided that the following conditions are satisfied:

1 Any directories inferior to the named directory are pseudo users' directories.

2 There is no job in progress in the directory that is to be erased.

When a user's proper directory is erased, the user is removed from the Dictionary and any outstanding budgets are transferred to its owner.

Examples

ERASE MYFILE(3)	Generation 3 of MYFILE will be erased providing the proper user has an ERASE trap for it.
ERASE :Z.TAB	The file TAB belonging to user :Z will be erased providing the proper user has been given an ERASE trap for it.
ERASE :OWE	:OWE is a pseudo user whose directory belongs to the proper user, so the directory will be erased if the proper user has an ERASE trap for it.

Error messages

z HAS A USER'S DIRECTORY INFERIOR TO IT

YOU MAY NOT ERASE AN ELEMENT OF A MULTIFILE

n JOBS IN PROGRESS OR UNACCOUNTED FOR IN DIRECTORY *z*

Note: It is possible for the count of jobs in progress or unaccounted for to be incorrect, thus preventing the directory being erased. Should the user suspect this is the case, he should inform the installation management who will take appropriate action (see Chapter 10 of the manual *GEORGE 3 and 4 Operation Management.* Edition 3, TP4310).

EXIT (EX)

Function

Causes the obeying of commands from the current macro definition file or job description file (that is, the file from which the EXIT command was read) to be terminated. This has the same effect as GOing to the end of the file.

Format

EXIT

The command has no parameters.

Forbidden contexts

None.

An EXIT command may only be issued from a macro definition file. Note that job description files are a subset of macro definition files.

Execution

If the source of command is not a file, an error is reported. Otherwise the file is closed and the message

END OF MACRO

is output to the monitoring file.

After this, one of the following may happen:

1 If the source of command at the next higher command processor level is found to be either a file which is not positioned within delimiters (see page 5) or something other than a file, then there is nothing more to be done and the command finishes.

2 If the source of command at the next higher command processor level is found to be a file which is positioned within delimiters, then a search is made for the stopper (recursively upwards if no stopper is found before the end of file). The next command to be read will be the one after the stopper. If no stopper is found but the next higher command processor level is not between delimiters, no further search is made. For every file that is closed in the search for a stopper, an

END OF MACRO

message is sent to the monitoring file.

Notes

The following notes apply only to off-line jobs:

1 If an EXIT command is issued from a job description file, this will result in the job being suspended for a length of time determined by the WAITTIME installation parameter. (See *INSTPARA* page 294). Thus, the effect is the same as for a job description file which has no ENDJOB command.

2 The following combination of circumstances will also have the same effect as 1(above).

(a) An EXIT command is issued from a macro definition file.

(b) At each higher command processor level the source of command is a file which is positioned within delimiters.

(c) No stopper is found before reaching the end of the job description file.

Error messages

IS ONLY ALLOWED FROM A MACRO DEFINITION FILE

EXPRESS (EP)

Function

1 Informs the high level scheduler subject program that a particular job is of great urgency and should not be made to wait before being made fully started.

2 Communicates a message to the high level scheduler subject program, which will cause it to perform one of a number of tasks.

Format

```
1   (a)  EXPRESS   job number
    (b)  EXPRESS   jobname,username

2   (a)  EXPRESS   job number,text
    (b)  EXPRESS   jobname,username,text
```

In Formats 1(a) and 2(a) *job number* is the number that the system uses to identify a particular job. The operator can determine this number by means of a WHATSTATE command.

In Formats 1(b) and 2(b) the *jobname* and *username* parameters are interchangeable.

In Formats 2, the *text* must be one of a set of character strings known to the high level scheduler, or the command will be ignored. The *text* may not be longer than 24 characters; excess characters are truncated.

In Format 2(a), the first parameter may be null. Thus in cases where the *text* is not applicable to a particular job, it is not necessary (although it is still possible) to refer to a job number.

Forbidden contexts

NOT OPERATOR

Execution

FORMAT 1

The command checks whether the specified job exists and reports an error if it does not. The EXPRESS command has no effect for a system issued job; if it is issued for such a job the following message is output to the operator's console:

```
EXPRESS COMMAND IS NOT MEANINGFUL FOR A SYSTEM ISSUED JOB
```

If the specified job exists and is not system issued, the high level scheduler subject program is informed that an EXPRESS command has been issued for this job and one of the following messages is output to the operator's console, depending on which format of the command was used.

If Format (a) was used:

EXPRESS COMMAND FOR JOBNUMBER *x* ACKNOWLEDGED

If Format (b) was used:

EXPRESS COMMAND FOR *jobname, username* ACKNOWLEDGED

The job is given a high Executive priority by the low level scheduler. The job is given UR A when it becomes fully started. No budget checking is done. If the job is already fully started, then the urgency cannot be reset unless the job becomes tentatively started and then fully started again.

If the EXPRESS command is cancelled, then the job will have its urgency allocated in the usual way.

For further details of how the high level scheduler makes jobs fully started see Chapter 8 of the manual *GEORGE 3 and 4 Operation Management.*

FORMAT 2

If the *text* is one of a set of character strings known to the high level scheduler, then appropriate action is taken; otherwise the command is ignored. The character strings for the standard ICL high level scheduler are defined in Chapter 8 of the manual *GEORGE 3 and 4 Operation Management* and they instruct the HLS to perform certain tasks. These tasks include BREAK, SWITCH, CANCEL EXPRESS, allow/inhibit job saving, allow/inhibit budget checking.

FILEIN

Function

Copies subfiles from a magnetic tape into files in the filestore.

Format

FILEIN   R *terminator*, T *terminator*, *CR or *TR *file description*, ER *label*, SF *label*

The FILEIN command causes a number of lines of steering information to be read either from the job source (the job description file or MOP terminal) or from a file. A maximum of 128 characters can be read in at one time. The steering lines have the basic format:

subfile name, file description

The subfile name is the name of the subfile that is to be copied into the file indicated by the file description. A subfile name has the format of a general local name. The file description may be a file name or a workfile name.

The subfile, however, may be specified by a number of levels, ending with the name of the required simple subfile.

e.g. A(3).B(2)

This refers to a simple subfile called B with a generation number of 2, which is immediately inferior to a composite subfile A with a generation number of 3. Generation 0 must be specified. If no generation number is given 'any' will be assumed.

Language code may also be specified, for example, A(3).B(2/A); if not given 'any' will again be assumed. If only one character is specified, the remainder of the word is space filled, that is, A becomes A∇∇∇.

A simple subfile to be copied in may be specified by its name alone, or by a series of names giving the higher level composite subfiles which contain the simple subfile. For example, if subfile A with file generation number 3 is to be copied in, it may be specified by:

A(3)

B.A(3)

or

X.T.Z.B.A(3)

One subfile may be copied into more than one, but not more than five files. The file names of those files into which the subfile is to be copied must follow the subfile description in the steering line. For example:

A(2).B(1),FILE1,FILE2,FILE3

If the subfile description and the file description(s) will not fit into one steering line, the file description(s) may be continued on more than one line. The continuation is indicated by the character '@' in the overflowing line. For example:

Line 1	SUBFILE 1(2/C). SUBFILE 2(3/B), FILE@
Line 2	1, FILE2, FILE3

There should be no spaces between the continuation character and the previous character of the file description unless they are actually part of the file description.

The subfile description, upto and including the first comma, must occupy the first part of the steering line.

If no file description is specified in the steering line, that is, if only a subfile description is specified, the subfile will be copied into a file with the same name as the simple subfile on the tape. For example:

EDIT

is the same as EDIT, EDIT

If no file description is given but the subfile description is followed by a comma, the steering line will be taken as an 'ignore' parameter. This may be used when there are subfiles with the same name on the tape and it is not required to copy them all in. The steering lines will be searched in the order in which they are read in; hence if an 'ignore' parameter is placed in the correct position within the steering lines, the required subfile will be ignored.

If it is required to copy more than one subfile to the same filestore file, that is to append information to an already existing file, (APPEND) should be included after the file description of the appropriate file. For example:

SUB1.SUB2, FILE(APPEND)

Note: If '(APPEND)' is not included, the file will be overwritten.

The subfile description may describe a composite subfile alone if the steering line is preceded by an asterisk. (In this case, all the simple subfiles within the composite subfile described are copied into files with the same names as the subfiles themselves.) For example:

*SUB1.SUB2

If the subfile description is followed by a comma the whole of the composite subfile described will be ignored.

All the parameters are optional and may occur in any sequence.

The steering lines are read in groups. The R terminator indicates the end of each group of parameters and the T terminator indicates the end of the complete run. This gives the user the option to copy in from more than one tape.

If the R *terminator* is specified, the R must be followed by the first non-space character of the line which terminates the steering information to be read in at one pass, and the three characters which follow. If it is not given, R#### will be assumed.

If the T *terminator* is specified, the T must be followed by the first non-space character of the line which terminates all the steering information, and the three characters which follow. If it is not given, T**** will be assumed.

The *CR or *TR *file description* parameter defines the source of the steering information as a card or paper tape file with the file description given. If it is omitted, or if only *CR or *TR is given, steering information will be read from the job source.

The ER *label* parameter defines a label to which control is to be passed in the event of a failure during the run. Though the parameter is optional, it is advisable to include it whenever FILEIN is issued from a job description file. If it is omitted, the next command after the FILEIN is obeyed if the run fails, i.e. the program deletes ER.

The SF *label* parameter defines a label to which control is to be passed if, at any time during the run of the program, an error has been encountered within the subfile being copied in and hence the program has deleted SF. If it is omitted, the next command after the FILEIN is obeyed.

The magnetic tape description to indicate the tape containing the subfiles to be copied, must be specified within each group of steering lines. The format of the tape description steering line is:

, *magnetic tape description*

The tape is made ONLINE to the program as soon as the steering line is read in. If a terminator is read and no tape description steering line has previously been read, the tape still ONLINE will be rewound and researched. Otherwise the tape will be closed and the next one ONLINEd as MT0.

If no steering lines precede an R terminator and this terminator is qualified by the characters NULL, the whole of the current tape will be copied in. For example:

####NULL

In this case, each simple subfile on the tape will be copied into a file in the filestore named after itself.

Forbidden contexts

NO USER, BREAK-IN

Execution

The FILEIN command is a system macro which loads and enters a subject program. If the steering lines are read from a file, that file is ASSIGNed to the program; otherwise the job source is made ONLINE to the program. Each line of steering information is stored within the program to be processed later. Shift characters ↑, [and $ occurring within a steering line are ignored. ↑* is taken to indicate the end of a line.

When the magnetic tape description steering line is read, it is written to the monitoring file system (DISPLAY category) and then the required tape is made ONLINE.

When a terminator is read, the required subfiles are copied in from the tape. Each steering line is written to the monitoring file system (DISPLAY category) before the specified subfile is copied in. Each subfile that is copied must consist of batched or unbatched records not more than 511 words long including the header, or binary non-overlay program blocks; the maximum block size is 512 words. If the end of tape is reached and all the subfiles specified in the steering information have not been found the DISPLAY message:

SUBFILES NOT ON TAPE

is sent and all the steering lines specifying subfiles not copied in are also DISPLAYed.

The program then DISPLAYs the current terminator, and reads in the next group of steering lines.

When all the required subfiles have been copied in, i.e. after the T terminator has been read, the program deletes OK, or SF if an error was encountered within a subfile being copied in.

Examples

```
FILEIN    R////
*SF1,
,PROGRAM∇TAPE
SF1.A,FILEA
////
FILEB.A
****
```

Subfile A within the second composite subfile SF1 on a tape called PROGRAM∇TAPE will be copied into a file called FILEA. The tape will then be rewound and subfile A within composite subfile FILEB will be copied into a file called A.

```
FILEIN
,:USER.M1 (3/200)
####NULL
****
```

The whole of the librarian tape :USER.M1 (3/200) will be copied into files in filestore.

```
FILEIN    *CR STEER
,TAPENAME
SUBC.A(99/B100),FILE
A(44/C100)
####
,(360135)
B.C.D,
B.C.D
****
```

The steering lines will be read from a card reader file called STEER. A tape called TAPENAME will be made ONLINE and subfile A with generation number 99 and language code B100 within composite subfile SUBC will be copied into a file in the filestore called FILE. Subfile A with generation number 44 and language code C100 will be copied into a file in the filestore with the same name, generation number and language code as the subfile. Tape TAPENAME will then be closed and tape 360135 made ONLINE. The first simple subfile D within composite subfile C within composite subfile B will be ignored and the second will be copied into a file called D.

```
FILEIN    TZZZZ
*DATA
A,FILEA,FILEB
,DATATAPE
ZZZZ
```

On a tape called DATATAPE all the simple subfiles within the composite subfile DATA will be copied into files of the same name. Also, Subfile A will be copied into FILEA and FILEB in the filestore.

```
FILEIN
A,B
,(361144)
ABC(9/B1),FILEA,FILEB,@
FILEC,FILED
####
*ABC
****
```

On tape 361144 Subfile A will be copied into a file called B and subfile ABC with generation number 9 and language code B1∇∇ will be copied into FILEA, FILEB, FILEC and FILED in the filestore. The tape will be rewound and the whole of composite subfile ABC will be copied in.

Error messages

ASSIGN and ONLINE error messages

The following are given as DISPLAY messages. A message that refers to an error in a steering line will in most cases be followed by the erroneous steering line in the display category of the monitoring file.

DISPLAY Message	*Reason*	*Subsequent action*
ERROR IN TERMINATOR PARAMETER		The program deletes ER.
FORMAT ERROR IN STEERING LINE	The subfile name has more than 12 characters or illegal characters in steering line.	The incorrect steering line is ignored and the program reads the next.
FORMAT ERROR IN SUBFILE NAME	The subfile name does not begin with a letter or contains illegal characters.	As above.
FORMAT ERROR ON TAPE	A trailer label, start or end of subfile sentinel has been read at an unexpected time.	The program deletes ER.
TAPE FAILURE		Program deletes ER.
TAPE NOT ON LINE	Command error occurs on issuing the ONLINE.	All steering lines up to terminator are ignored.
FILE NOT ASSIGNED SUBFILE NOT COPIED IN	Command error occurs on ASSIGNing the only file to be copied into	The program continues without copying in this subfile.
FGN GREATER THAN 4095	A subfile with file generation number greater than 4095 has been specified in a steering line.	The steering line is ignored and the program reads the next.
MAGNETIC TAPE DESCRIPTION MISSING	No tape description steering line has been read in the first group of steering lines.	All steering lines up to the terminator are ignored.
SUPERFLUOUS COMMA IN STEERING LINE	The last file description in the steering line is followed by a comma and no continuation character.	The comma is ignored and the program continues normally.
MORE THAN 5 FILE DESCRIPTIONS SPECIFIED	A steering line specifying more than 5 file descriptions for the subfile to be copied into, has been read.	The program ASSIGNs the first 5 files and the rest are ignored.
TAPE ALREADY ON-LINE	More than 1 tape description steering line has been read within a group of parameters.	The steering line is ignored and the program continues.

DISPLAY Message	*Reason*	*Subsequent action*
FORMAT ERROR IN SUBFILE	The contents of the subfile being copied in, are not in standard format.	The program continues leaving the contents of the current output file(s) undefined.
BUFFER FULL	The program's storage area is full and no more steering lines can be read in at this pass.	The program processes all the steering lines read in and then reads in the remainder.
SUBFILE NESTED TO MORE THAN 6 LEVELS	(a) A steering line has been read with a subfile description specifying more than 6 levels.	(a) The steering line is ignored and the program continues by reading in the next.
	(b) A subfile nested to more than 6 levels has been read on the input tape.	(b) The program skips down the tape until the level count is less than 6.
NO STEERING LINES READ	An unqualified S terminator has been read.	The program continues by reading in the next group of steering lines.

FIND

Function

Loads the named program from the specified magnetic tape or UDAS device, or from a filestore magnetic tape file.

Format

1 To load a program from a specified magnetic tape or UDAS device:

FIND *program name,* $\left\{\begin{matrix}\text{MT}\\\text{DA}\end{matrix}\right\}$*,entrant description,* $\left\{\begin{matrix}\text{CORE}\\\text{SIZE}\end{matrix}\right\}$ *number,* OVERLAYS *magnetic tape description*

2 To load a program from a filestore magnetic tape file:

FIND *program name,, file description,* $\left\{\begin{matrix}\text{CORE}\\\text{SIZE}\end{matrix}\right\}$ *number,* OVERLAYS *magnetic tape file description*

The *program name* has the standard format of # followed by four characters. This parameter is mandatory.

The *device type* is MT if the program is to be loaded from magnetic tape, or DA if it is to be loaded from a UDAS device (Format 1 above). In order to retain compatibility with preceding marks of GEORGE, ED or FD may be used, for the time being, instead of DA.

If the program is to be found from a filestore magnetic tape file, the second parameter must be null (Format 2 above). It must not be omitted.

The *entrant description* specifies the magnetic tape or UDAS device from which the named program is to be loaded (Format 1 above). Accordingly, this parameter is a magnetic tape description or an exofile description as appropriate.

Note that the magnetic tape or 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.

If format 2 of FIND is used, the third parameter is a *file description* specifying the magnetic tape file from which the named program is to be loaded.

The third parameter is mandatory in both formats.

The CORE or SIZE and OVERLAYS parameters are optional and may occur in either sequence. The CORE or SIZE parameter is equivalent to the CORE command (see page 242) or the SIZE command (see page 400.2) and specifies the size of the program to be loaded.

The OVERLAYS parameter is needed only if the program being loaded is an overlay program in consolidated semi-compiled format (the format produced by magnetic tape compilers). In this case an output tape is needed, to which the General Purpose Loader will write the binary program. The magnetic tape description for this tape must be specified by the OVERLAYS parameter. The OVERLAYS parameter may be given with a magnetic tape description if the program is being loaded from magnetic tape, or with a magnetic tape file description or workfile if the program is being loaded from a filestore magnetic tape file. If the OVERLAYS parameter is given on its own, a magnetic tape workfile will be created to hold the binary program.

Notes:

1 The process of loading a program in consolidated semi-compiled format is inefficient and should be avoided when possible. For overlay programs this can be accomplished by using the binary form of the program (the tape or file specified in the OVERLAYS parameter) for all subsequent loading. Thus, there is seldom a need for the workfile option of the OVERLAYS parameter, unless of course it is known that the program is not to be preserved. For non-overlay programs, the program can be SAVEd immediately after loading and loaded subsequently from the saved file.

2 If the *third* parameter should happen to consist of the characters CORE, SIZE or OVERLAYS, then it must be enclosed in parentheses.

Forbidden contexts

NO USER, BREAKIN, PROGRAM

After the command, the context will be CORE IMAGE if the specified program has been loaded, or NO CORE IMAGE if the specified program cannot be found.

Execution

FIND is a macro that loads a bootstrap (a subject program), makes the specified device ONLINE (Format 1) or ASSIGNs magnetic tape to the specified file (Format 2), and enters the bootstrap which searches for the specified program. When the program is found the message:

PROGRAM *program name* FOUND

is displayed.

The program's request slip is examined and the requested core obtained. (If the CORE or SIZE parameter is present, the number given there is taken to indicate the size required.) The program's request slip is then given to GEORGE and the program loaded.

Notes:

1 If the program has a type 6 supplementary request block, an attempt will be made to obtain the modes required by means of a 165/9 extracode.

2 If the program has a type 2 entry block, the program will be entered at the specified address only if the GPL bit in the request slip is set. Otherwise the bootstrap will HALT GO without entering the program, as in the LOAD command. This is different from previous marks, and any installation seriously embarrassed by the difference should edit IF HALTED GO, RESUME at the end of the macro.

3 If the program being loaded has no binary block with a destination address in the range 0 to 7 and if the program event previous to the FIND command was caused by a 160/2 extracode, the values in the accumulators of the core image will be the same after the command as before the 160/2 extracode was issued.

4 FIND accesses the entrants :LIB.PROGRAM∇XK62,:LIB.PROGRAM∇XK73, which is used by GEORGE 3, and :LIB.PROGRAM∇XK74, which is used by GEORGE 4. All three entrants are accessed in EXECUTE mode.

5 Under GEORGE 4, :LIB.PROGRAM∇XK74 will load a sparse program provided the highest destination address specified in its binary block is not in the range (4M-64K) to 4M.

6 Although :LIB.PROGRAM∇XK74 will set appropriate permissions if special permission areas are specified, no account will be taken of shareable areas indicated in the supplementary request slip.

7 The user is warned that at present, if he breaks in during the execution of the FIND macro in such a manner as to overwrite the current display, the macro may fail to function correctly.

Examples

FIND #XE20,DA,(775173,PROGRAM∇XE20),- CORE 22000	#XE20 will be loaded from the UDAS exofile with cartridge number 775173 (octal) and exofile name PROGRAM∇XE20, and will be given 22000 words of core.
FIND #XK99,,PROGRAM∇TAPE, OVERLAYS	#XK99 will be loaded from the magnetic tape file PROGRAM∇TAPE, and a workfile will be created to hold the overlays.

Error messages

The error messages given below may be sent to the monitoring file in the DISPLAY category. In the case of the first two messages described, the context after the FIND command will be the same as when the command was issued. In all other cases the context after the command will be NO CORE IMAGE.

Message	*Reason*
PARAMETER 1 NOT RECOGNISED	Parameter 1 does not begin with #.
PARAMETER 2 NOT RECOGNISED	Parameter 2 is neither MT, DA, ED, FD or null.
device description NOT AVAILABLE	Either there is an error in the third parameter or the operator has indicated that the specified device is not available in response to a system request to load it.
file description NOT AVAILABLE	There is an error in the third parameter, when Format 2 has been used.
ERROR IN *program name* 'S REQUEST SLIP	The request slip of the program has been checked and an error found.
device type FAIL OR BINARY PROGRAM FORMAT ERROR	The specified device type has failed; in the case of magnetic tape this could be caused by a non-binary block.
device type READ OR WRITE ERROR	There has been a read or write failure on a UDAS device transfer, or a read failure on a magnetic tape.
PROGRAM *program name* NOT FOUND	The program cannot be found on the specified device or in the specified magnetic tape file.
ONLINE $\left\{\begin{matrix}\text{MT}\\\text{DA}\end{matrix}\right\}$ FAIL	The specified device has failed.
DA BLOCK FORMAT OR CHECKSUM ERRORS	This message occurs only if the device type is ED, FD or DA
22AM/EBM NOT AVAILABLE	The program requires either 22AM, EBM or both but the required mode could not be obtained.
magnetic tape description NOT AVAILABLE	There is an error in the magnetic tape description given with the OVERLAYS parameter, or the operator has indicated in response to a system request to load it, that the specified device is not available.
UNIDENTIFIED ERROR	Some unknown error has occurred.
PROGRAM EXCEEDS REQUESTED SIZE	The program is larger than the size specified in the program's request slip or by the CORE or SIZE parameter.
PROGRAM IS TOO BIG	The program size required exceeds the installation parameter COREOBJECT or a MAXSIZE command, or the highest destination address is in the range (4M-64K) to 4M.
program name IS SPARSE FIND UNDER GEORGE 4 ONLY	An attempt has been made to load a sparse program under GEORGE 3.

FINISH (FN)

Function

Prevents the system from accepting or starting any more jobs. If a parameter is given, existing jobs will be finished immediately or within the time specified.

A description of CANCEL FINISH follows this specification.

Format

1 FINISH NOW

2 FINISH *length of time*

If no units are given, MINS is assumed; the parameter has the same format as the milltime parameter (see page 164) except that HRS (hours) is also allowed. The maximum value for this parameter is 72 hours.

3 FINISH

Forbidden contexts

NOT OPERATOR, REMOTE

Execution

No more jobs will be accepted by the system and a user who tries to introduce a new job will receive the message:

$$\text{NO MORE JOBS ARE BEING ACCEPTED: SYSTEM CLOSING DOWN} \left\{ \begin{array}{l} \text{NOW (Format 1)} \\ \text{IN } \textit{time} \text{ (Format 2)} \\ \text{SOON (Format 3)} \end{array} \right.$$

Further action taken depends on the format, as follows:

FORMAT 1

The High Level Scheduler program will be informed so that it may take appropriate action and no more jobs will be made fully started. Any jobs already in the system will be abandoned. The message:

ABANDONED : SYSTEM CLOSING DOWN

is output to the MOP terminal and/or the monitoring file. All LISTFILEs and INPUTs in progress will be abandoned. When all user jobs have finished or been abandoned, the message

ALL USER JOBS NOW FINISHED

is output on the operator's console.

FORMATS 2 AND 3

All MOP jobs in progress will be informed with the message

$$\text{SYSTEM CLOSING DOWN} \left\{ \begin{array}{l} \text{IN } \textit{time} \text{ (Format 2)} \\ \text{SOON (Format 3)} \end{array} \right.$$

The High Level Scheduler program will be informed and no more jobs will be made fully started. Tentatively started jobs will be run until they have finished or require to be made fully started. In this case background jobs will be abandoned with the message

ABANDONED: SYSTEM CLOSING DOWN

and MOP jobs will receive the message:

YOU MAY NOT BE MADE FULLY STARTED

Fully started background jobs will be allowed to continue until either they have finished or (in format 2) the time given as a parameter has elapsed, when FINISH NOW action will be taken for them. Background jobs and MOP jobs in limbo may be connected until the *time* specified expires. LISTFILEs waiting to be output can be initiated provided (in format 2) that this takes place before the time given has passed. At the end of this time any LISTFILEs and INPUTs in the above categories will be allowed to finish. LISTFILEs to the monitoring file and INPUTs in USER context will be abandoned when the job issuing them is abandoned.

When all background jobs have finished, the message

ALL BACKGROUND JOBS FINISHED: *n* MOP JOBS LEFT

is output on the operator's console (provided it is before *time* in format 2). When all jobs, LISTFILEs and INPUTs have finished, the message

ALL USER JOBS NOW FINISHED

is output.

At this stage, system-issued jobs may still be running. When these too have terminated, the message

EVERYTHING HAS FINISHED

will be output and the system can be deleted.

Notes

1 A user's job may take its own FINISH action. If the job has previously issued a WHENEVER FINISH command (see page 400.30) then the command given in the second half of the WHENEVER command will be obeyed. Such jobs can also test whether a FINISH command has been issued by means of the IF FINISH command (see page 287). Note that this condition is unset by the CANCEL FINISH command (see below).

2 System issued jobs must take their own FINISH action. They can detect FINISH by using WHENEVER FINISH and IF FINISH in the same way as user jobs. They can also use a FINISHED qualifier in the same commands, and this becomes true when all user jobs have finished. For the dumper's action on FINISH, see Chapter 4 of the manual *GEORGE 3 and 4 Operation Management*, TP4334.

3 The length of time may be extended or shortened by issuing another FINISH command provided that this occurs before FINISH NOW action has been taken.

4 The effect of a FINISH command may be cancelled by use of the CANCEL command (see below).

5 Any jobs which have been saved by the high level scheduler will remain saved over the system close down if there is no general restore at E.M.S.

Error messages

PARAMETER FORMAT ERROR

FINISH NOW ACTION ALREADY STARTED

CANCELLATION OF FINISH

Function

Cancels the effect of a FINISH command.

Format

CANCEL FINISH

Forbidden contexts

NOT OPERATOR, REMOTE

Execution

Provided that FINISH NOW action has not already been initiated, the ban on accepting jobs will be lifted and they may be introduced subject to the usual JOBLIMIT restrictions. Any MOP jobs still running will be informed of the CANCEL with the message:

SYSTEM CLOSE DOWN HAS BEEN CANCELLED

Any jobs already abandoned must be restarted by the operators. Any LISTFILE or INPUT commands not permitted after FINISH must be reissued.

Error messages

ONLY ALLOWED AFTER FINISH

FINISH NOW ACTION ALREADY STARTED

FULLTRACE (FT)

Function

Causes all relevant categories of monitoring file output to be sent to the job's monitoring file, and causes all subsequent TRACE commands in the job to be ignored.

Format

```
        FULLTRACE
```

Forbidden contexts

NO USER

Execution

The system is informed that all future TRACE commands should be ignored, and that all possible categories of messages must be sent to the job's monitoring file. This effect will hold until the end of the job is reached.

Example

```
FULLTRACE
```

Error messages

None

GET (GE)

Function

Causes a magnetic tape currently in the pool to be allocated to the user or an entry for a named worktape to be set up in the job's temporary directory (see *Acquiring worktapes,* page 73).

Format

1 GET *magnetic tape name*

The magnetic tape name must be qualified by *MT and must be relative to the user's proper directory or to a pseudo user's directory owned by the user (unless the user is suitably privileged, see *Ownership of entrants,* page 175).

The entrant description qualifiers FROZEN, OWNERACC, TRAPGO, TRAPSTOP may be given with the GET command, see *Optional entrant description qualifiers,* page 170, for details

2 GET *worktape name*

The worktape name must be qualified by *MT

In either format the PR *magnetic tape properties* entrant description qualifier may also be given to obtain a tape with particular characteristics. More than one magnetic tape property may be specified; only tape characteristic properties or MODE *n* are significant.

Forbidden contexts

NO USER

Execution

For both formats a tape with the specified characteristics (if any were given) will be obtained. If no characteristic was specified then a tape with the standard characteristic will be obtained. If there is no standard then any available pool tape (Format 1) or worktape (Format 2) will be used.

FORMAT 1

A test is made to ensure that the user's budget of SPACEMT permits the addition of another tape. A request to the operator to load a pool tape is then generated. When one is loaded, an entry for this tape is set up in the directory implied by the magnetic tape name, and the user name associated with this directory is entered in :SYSTEM.SERIAL to indicate that the tape is an owned tape. The specified local name of the tape is written in a new header label. The following message is output on the operator's console:

MT *tape serial number* IS NO LONGER A POOL TAPE

and the message

USED MT *tape serial number, magnetic tape description*

is sent to the monitoring file system (COMMENT category)

READ, WRITE and ERASE access traps are set up for the owner and his SPACEMT budget is updated.

FORMAT 2

A worktape is obtained and left on a deck ready for use, and an entry is made in the job's temporary directory. The name in the tape's header label is not changed and so still contains whatever was there when the tape was last used. The tape has a write permit ring.

Example

```
GET   MTDATA (*MT)
GET   !WORKTAPE (*MT)
GET   REGA(*MT,PR NRZI)
```

Error messages

DEVICE TYPE QUALIFIER IMPERMISSIBLE

DEVICE TYPE QUALIFIER MISSING

MAGNETIC TAPE DESCRIPTION FOR THIS COMMAND MUST CONTAIN A LOCAL NAME

MAGNETIC TAPE DESCRIPTION FOR THIS COMMAND MUST NOT CONTAIN A SERIAL NUMBER

NO POOL TAPE AVAILABLE

USER HAS NO SPACEMT BUDGET AVAILABLE

USER IS OVERDRAWN ON SPACEMT BUDGET

NO WORKTAPE AVAILABLE

ILLEGAL COMBINATION OF PROPERTIES

ILLEGAL MODE GIVEN

NO SUITABLE DECK AVAILABLE

GETONLINE (GL)

Function

Takes a magnetic tape from the pool, relabels it and makes it on-line to a program, or takes a worktape and makes it on-line to a program as a named worktape.

Format

1 GETONLINE *magnetic tape peripheral name, magnetic tape name,* PR *peripheral property names*

The first parameter is the peripheral name specified in the program's PERI instructions. It may be qualified by READ or WRITE; if no qualifier is given, WRITE is assumed. The second parameter is the name that is to be given to the tape. It must be relative to the user's proper directory or to a pseudo user's directory owned by the proper user (unless the user is suitably privileged, see *Ownership of entrants,* page 175). For the use of signed generation numbers see *Magnetic tape names,* page 168. The entrant description qualifiers FROZEN, OWNERACC, TRAPGO, TRAPSTOP may be given with the GETONLINE command (see *Optional entrant description qualifiers,* page 170). The third parameter is optional but may be used to specify any special deck characteristic or speed

2 GETONLINE *magnetic tape peripheral name, worktape name,* PR *peripheral property names*

The first parameter is the peripheral name specified in the program's PERI instructions. It may be qualified by either READ or WRITE. If no qualifier is given, WRITE is assumed.

In either format the PR *magnetic tape properties* entrant description qualifier may also be given to obtain a tape with particular characteristics.

Forbidden contexts

NO CORE IMAGE

Execution

FORMAT 1

GETONLINE is a command that combines the functions of GET and ONLINE. Provided that the user's budgets permit, a permanent directory entry for the magnetic tape is set up in the directory indicated by the magnetic tape name. A pool tape is then loaded, given a new header label and made on-line to the program. Initial user traps are set up as in the GET command.

FORMAT 2

A worktape is obtained and made on-line to the object program and an entry is made in the job's temporary directory (see *Worktapes,* page 76). The name in the tape's header label is not to be altered and so contains whatever was there when the tape was last used. The tape has a write permit ring.

For both formats a tape will be obtained with the characteristics specified in the PROPERTY qualifier. Any characteristics specified in the PROPERTY parameter will also be taken into account. If no characteristic was specified a tape with the standard characteristic will be obtained. If there is no standard then any available pool tape (Format 1) or worktape (Format 2) will be used.

Examples

GETONLINE *MT1,:PSEUDO-1.TAPEDATA	:PSEUDO-1 must be a pseudo user whose directory belongs to the proper user. TAPEDATA is the name that is given to the pool tape. *MT1 is the program channel to which the tape will be ONLINE.
GETONLINE *MT0,!WORK1(PR MODE20),PR FAST	A 7-track worktape in mode 20 will be obtained on a fast deck. It will be given the worktape name !WORK1, and connected to the program channel *MT0.

Note

GETONLINE is the command equivalent of the open mode PERI #400.

Error messages

As for the GET and ONLINE commands.

GOTO or GO TO (GO)

Function

Causes a branch to be made to another (labelled) command in a job description or macro definition file.

Format

GOTO *label*

or

GO TO *label*

where GO and TO may be separated by one or more spaces. The label may be of any length and must start with a digit.

Forbidden contexts

None

A GO TO command may only be issued from a job description file or macro definition file.

Execution

Either the job description or macro definition file is scanned for the specified label, ignoring all lines between delimiters except the first. If the GOTO command itself is between delimiters, the search starts from the command after the stopper: otherwise the search begins at the command following the GOTO command. If the end of the file is reached without the label having been found, the search starts again from the beginning of the file and continues up to the point at which the GOTO command was issued. If the label is not found, the file (if any) at the command processor level one higher is searched in the same way, but using the delimiters set at the new level. If the file at the higher level is positioned between delimiters, the GOTO command skips to the stopper before starting the search. Otherwise the search begins at the command after the macro command.

This process may need to be repeated for each macro level and job description file level (for off-line jobs), until an internally issued command, a program issued command or the top command processor level is reached. If the label is found, the reading and processing of the command continues from there, and all intermediate levels are closed. If, however, the label is not found then a command error is reported at the level of the GOTO command.

Notes:

1 For an explanation of command processor levels, see page 4.

2 If there are two identical labels within a single macro definition or job description the search will stop at the first one encountered.

3 It is not advisable to begin a continuation line with a numeral since the GOTO command will treat this as a label. This can also be avoided, by putting delimiters around multi-record commands (see page 5).

4 Command delimiters may be reset by the BRACKETS command (see page 218).

5 A current delimiter should not be used as the terminator of an INPUT command or inside embedded data, as this will interfere with GOTO commands.

Example

```
GOTO    1ABC
```

Error messages

IS ONLY ALLOWED IN A MACRO DEFINITION FILE

THE LABEL DOES NOT START WITH A DIGIT

LABEL PARAMETER MISSING

WE CANNOT FIND THIS LABEL