OPEN$ - Open Directory

The OPEN$ routine is available to open a GSM directory for subsequent processing by the LIST$ and ELIST$ functions.

1. Invocation

To open a directory for subsequent processing code:

```
CALL OPEN$ USING filename
```

where filename is the name of a closed FD.

2. STOP Codes and Exception Conditions

No STOP codes are generated by OPEN$.

The following EXIT codes may be returned by OPEN$:

EXIT code	**$$COND**	**Description**
12201	1	An irrecoverable I/O error has occurred.
12202	2	An attempt has been made to open a unit that does not contain a volume with a Global System Manager directory ($$RES = 6), or if it is in use as a spool unit ($$RES = 5). If either exception occurs the OPEN$ does not take effect and the file definition remains unchanged.

3. Programming Notes

The family of routines OPEN$, OPENS$, LIST$, ELIST$ and CLOSE$ can be used to determine the file-id and type of each file present on a Global System Manager direct access volume. OPENS$, rather than OPEN$, must be used if the unit is in use as a spool unit.

The OPEN$, OPENS$ LIST$, ELIST$ and CLOSE$ routines all require a filename as their first parameter. This name identifies an FD to be used in the directory processing operation. At a minimum you must code the FD statement and the following ASSIGN statement:

```
FD filename ORGANISATION organisation
ASSIGN TO UNIT unit-id FILE file-id [VOLUME volume-id]
```

You can use any convenient organisation (e.g. UNDEFINED, RELATIVE-SEQUENTIAL) since the one specified is immaterial as far as the directory routine is concerned. You should specify the file-id as a symbol, since LIST$ returns each file-id found to be present in this field. The volume-id should be specified as a symbol if you wish to examine it following a call of OPEN$.

The FD must be closed when it is passed to OPEN$. It will then be opened so that it can be processed by LIST$ or CLOSE$: the type of open is special, however, and prevents the FD from being used by any other file processing operation such as a READ or WRITE. When you have finished examining the directory you must close the FD using CLOSE$. It is then returned to the normal closed state and can, should you so wish, be processed by a conventional access method OPEN statement.

If no exception is returned the volume-id of the currently mounted volume will have been placed in the field you have specified using the FD. Note that this means that if the volume-id is returned and the file subsequently opened normally, then volume-id checking will take place unless you zeroise the volume-id field.

If you attempt an OPEN$ operation on an FD which is already open, either due to an access method OPEN statement or to a previous invocation of OPEN$, your program will be terminated with a stop code.

Note that the OPENS$ routine behaves in the same way as OPEN$ except that it will function on a spool unit (that is it will not return exception condition 2 ($$RES = 6)).

Directory operations are relatively slow, so whenever possible you should use conventional access method open operations to determine whether or not files are present. For example, to check whether a file of known type is present it is usually best to issue an OPEN OLD or OPEN SHARED for it.

The directory operations are best employed in applications which are not performance critical, such as displaying or listing file information in response to an operator enquiry, or in printing or conversion operations where a number of files on the same volume are subjected to lengthy processing. In both these cases the time spent accessing the directory is short in comparison with the display or file processing time.

4. Examples

[EXAMPLE REQUIRED]

5. Copy-Books

None.

6. See Also

OPENS$	Open Spool Directory
LIST$	List Directory
ELIST$	List Directory Extended
CLOSE$	Close Directory