COPYS$ - Copy Open Shared File

The traditional COPY$ sub-routine, described in the File Management Manual, cannot copy a file that is Open Shared. The COPYS$ routine, functionality **similar** to COPY$, does not suffer from this restriction.

1. Invocation

To copy an Open Shared file code:

CALL COPYS$ USING *filename-i filename-o [flag]*

where *filename-i* is the name of the file definition for the input file, *filename-o* is the name of the file definition for the output file; and *flag* is an optional PIC 9(4) COMP variable, or literal, is a flag indicating whether any outstanding locks on the source file are to be retained.

The filename that is associated with *filename-I* must be either closed or Open Shared when COPYS$ is called and it will remain either closed or Open Shared when the routine returns control. The filename that is associated with *filename-o* must be closed when COPYS$ is called and will remain closed when the routine returns control. If volume identification checking is specified by the VOLUME phrase appearing in either or both FDs, it will be applied in the normal way when the copy routine opens the file for processing.

If a file with the same name as the output file already exists on the output volume, it will be deleted. The output file is always allocated anew. It is allocated the extent specified in its file definition's SIZE statement except that if the SIZE statement is omitted, or the actual size is specified as zero, the input file extent size will be used.

The file definition for the *input file* need contain only the FD and ASSIGN statements.

The file definition for the *output file* must contain the FD and ASSIGN statements and have the same ORGANISATION phrase as the input file. You may, if you wish, code ORGANISATION UNDEFINED in both cases and possibly avoid an unnecessary access method routine being included in your program. Unless you supply a SIZE statement and establish a non-zero size the file will be allocated the same size extent as the input file. Other file definition statements are unnecessary, since any additional file attributes are taken from the input file.

The optional flag parameter can be either:

0 Any outstanding locks on *filename-i* will be removed
1 Any outstanding locks on *filename-i* will be retained

2. STOP Codes and Exception Conditions

No STOP codes are generated by COPYS$.

The following EXIT codes may be returned by COPYS$:

EXIT code	**$$COND**	**Description**
7601	1	An irrecoverable I/O error occurred on either the input or output file.

7902	2	Exception condition 2 may be generated for a variety of reasons, identified by the value of $$RES: ● Input file not available ($$RES="3"); ● Output file capacity less than that of the input file ($$RES="5").

3. Programming Notes

COPYS$ is only available with GSM SP-25, or later.

The COPYS$ system routine copies an input file to an output file without performing any conversion process. It employs a temporary 32Kb memory page. The routine therefore executes much faster than the conversion routine which has to process a record at a time. On the other hand COPYS$ **cannot** be used to change indexed sequential to relative sequential or drop logically deleted records. However, by copying an indexed sequential file to a larger area its overflow area will be automatically extended without the overhead of reorganising the file.

COPYS$ can be used on direct access files but should not be employed to transfer data to a printer since this will not result in a properly formatted report appearing. The routine is not concerned with record format and can be used to transfer any sort of information.

If the size specified is 999999999 the output file extent will be allocated the maximum amount of contiguous space available.

When control is returned following an exception both FD's will be closed in the normal way. An incomplete output file may be in existence in this case. If so it can be deleted either by using the file utility command program interactively or by using the copy utility again after taking corrective action.

The size of the output file extent must be at least as great as the extent occupied by the input file. If you wish to reduce the size of the extent occupied by a direct access file you cannot use COPYS$. Instead you must use an OPEN OLD followed by a CLOSE TRUNCATE.

When the output file size is greater than that of the input file the extra space is initialised to binary zeros. This automatically extends the overflow area associated with an indexed sequential file.

4. Examples

[EXAMPLE REQUIRED]

5. Copy-Books

None.

6. See Also

COPY$ File copy routine
COPYP$ Copy password protected file