# LIBR$ - Copy Library Index Routine

The LIBR$ system routine is used to build an index of the names and starting positions of the books in a Global Cobol copy library. Subsequently individual books can be accessed using the text file access method by performing a READ statement followed by READ NEXT statements.

## 1. Invocation

To obtain a Copy Library index code:

```
CALL LIBR$ USING fd li [fm]
```

Where *fd* is the FD of the text library file, which must have Text File organisation and must be closed when the routine is called, and will remain closed when the routine returns control.

The parameter *li* is the name of a control block in which the index will be built, and consists of: the name table; a filler to hold the terminating zero byte if the library contains the maximum of 100 books; and the start table. The format is:

```
01    LI
 03   LINAME OCCURS 100 PIC X(2)          * NAME TABLE
 03   FILLER              PIC X                  * MAX. TERMINATOR
 03   LISTART OCCURS 100 PIC 9(6) COMP* START TABLE
```

When LIBR$ returns control it will have placed the names of the books the library contains in the name table in the order in which they were found. A corresponding entry in the start table will indicate the starting character number of the first line following the book start line. The last entry returned in the name table will be terminated by a binary zero byte so that a Global Cobol SEARCH can be employed to rapidly check for the presence of a book name.

The optional third parameter, *fm*, can be provided if spare memory is available and it is desired to optimise the performance of LIBR$ by supplying it with a large additional work area which it can use when scanning the text file. When the third parameter is used the quantity *fm* should label a free memory request block which has been previously set up to address an area acquired by the FREEX$ system routine. The block is therefore of the form:

```
01    FM
 03   FMFUN     PIC 9 COMP          * WILL CONTAIN 2
                                    * I.E. GET WORK SPACE
 03   FMSIZE    PIC 9(6)            * SIZE REQUIRED
 03   FMPTR     PIC PTR             * POINTER TO SPACE
 03   FMNAME    PIC X(8)            * NAME OF DATA PAGE
```

If an FM area is not supplied, or if the space actually allocated in FMALL is less then 512 bytes, LIBR$ will simply use the 512-byte FD extension as its work area and no performance optimisation will take place.

## 2. STOP Codes and Exception Conditions

No STOP codes are generated by LIBR$:

The following EXIT codes may be returned by LIBR$:

| EXIT code | $$COND | Description |
|---|---|---|
| 2101 | 1 | An irrecoverable I/O error has occurred while scanning the library file. |
| 2102 | 2 | The file cannot be found, or does not have Text File organisation. |
| 2103 | 3 | The text file is not in correct copy library format (for example, a book end line is missing) or a 101st book is encountered. A message giving the name of the book in error will be displayed on the screen. When control is returned following an exception the FD will be closed and the index table will contain a valid index to the part of the file already processed prior to the exception being detected. |

# 3. Programming Notes

The copy library is scanned sequentially. It is checked to be of the correct format, as described in Appendix F of the Global Cobol Language Manual. Each time a book start line is found a count is incremented by one and used to index the name table and the start table. The book name is placed in the name table and the character number of the line following the book start line is placed in the start table.

Once the library index has been constructed you read the lines of a particular book as follows:

- Use the Global Cobol SEARCH statement to find the index of the book name within the name table;
- Move the correspondingly indexed start table entry to the key field of the text file FD used to access the library;
- READ the first line of the book (i.e. the line immediately following the book start line);
- Obtain the second and subsequent lines of the book using READ NEXT.

The book is delimited by the book end line. Unlike any other line within the book, the first significant character of the book end line is an ASCII full stop. Thus to check for the book end line the simplest method is to redefine your record area as a table of single-character entries and check the entry whose number is returned in the TXSIG field to see whether it contains a full stop.

Note that a copy library may contain a null book, consisting of just a book start line immediately followed by a book end line. In this case the READ statement used to access the very first line of the book will obtain the book end line, and, in consequence, subsequent READ NEXTs will not be required.

# 4. Examples

[EXAMPLES REQUIRED]

# 5. Copy-Books

See copy-book "f$" in copy-library S.SYS32. Note that this copy-book **MUST** be expanded using a SUBSTITUTING clause. For example:

```
COPY "f$" USING "FM"
```

# 6. See Also

LIBRA$ Extended Copy Library Index Routine
FREEX$ Allocate 32-bit memory