Modular Map Input Output (MIO)

Eric Graham

July 7, 1982
Revised: July 8, 1982
Revised: August 1, 1982

SUMMARY OF REVISIONS:

PURPOSE:

The purpose of MIO is to simplify the application programs that access maps on the PDP 11s and to facilitate changing the actual structure of maps and their associated catalogs.

The MIO routines will hide as much as possible from the application programs. For example, the application program cannot tell in what format the map is stored on disk, scaled integer, floating point, etc. The application program cannot tell whether headers and map arrays are stored in one or more files.

The design of MIO must enable an implementation with a high overall I/O speed, however operations such as opening and closing maps, and setting or accessing header data may be relatively slow, since these operations are only performed a few times. In addition the implementation must be economical in address space and allow efficient use of overlays.

Both IMPS and AIPS have map I/O packages, there are overwhelming reasons for not adopting either system in its entirety, but we will unashamedly borrow from either whenever convenient.

MAPS:

A map is a two or three dimensional array of data with some additional data that we will call a header. If maps are to exist beyond the duration of a single task, they are catalogued. There is one catalog for each user.

HEADERS:

The header is any data in a map that is not part of the map array itself. Each field in the header may be accessed or modified.

There are certain items, which may be accessed as header data, but which are not saved at the time a map is closed. These are called pseudo-header data.

LOGICAL FUNCTION OPFMIO(MID,USER)
LOGICAL FUNCTION OPNMIO(MID)

OPFMIO opens the first map in the catalog of a specified USER. It returns a FALSE value if the user's catalog is empty. OPNMIO closes the map associated with MID and opens the next map after MID in the same users catalog. If no other map is found, then FALSE is returned. Remember that opening a map does not imply that any of the files containing map array data are opened or read.

SUBROUTINE LUMIO(LUN1,LUN2)

LUMIO may be used to specify a range of logical unit numbers that can be used by the MIO package. One logical unit number is required for each map that is open at any given time. LUMIO must be called before any other MIO subroutines. Subsequent calls to LUMIO have no effect.

SUBROUTINE GLMIO(LUN)
SUBROUTINE RLMIO(LUN)

If a logical unit number is required for non-MIO i/o operations, then the subroutine GLMIO may be used to get the value of a logical unit number from the pool that is managed by MIO. After the application program is finished with the logical unit number, it may be released for further use by RLMIO.

SUBROUTINE IDMIO(USER)

IDMIO may be called at any time to specify a default user number. If OPMIO does not specify a user explicitly in the NAME string, then the default is used.

SUBROUTINE DEMIO(MID)

Delete the map (and its disk files) and remove it from the users catalog.

SUBROUTINE UERMIO(ICODE)
SUBROUTINE ERMIO(ICODE)

The subroutine UERMIO is called by the MIO subroutines if an error occurs. ICODE is a two byte integer. The least significant byte specifies what type of an error occurred, RSX/11 I/O error codes are used for system errors. The most significant byte specifies where and in which MIO routine, the error was detected. The MIO subroutines call UERMIO before returning to the calling subroutine.

UERMIO is not supplied with the MIO package, it must be supplied by the application program. A subroutine called ERMIO is supplied which causes execution of the task to cease, and the value of ICODE to be printed. A simple version of UERMIO can be written which calls ERMIO. Alternatively a version of UERMIO may be written to perform any desired error handling. For example, the error code may be stored in a common block variable to be tested after the MIO routine returns.

LOGICAL FUNCTION OPFMIO(MID,USER)
LOGICAL FUNCTION OPNMIO(MID)

OPFMIO opens the first map in the catalog of a specified USER. It returns a FALSE value if the user's catalog is empty. OPNMIO closes the map associated with MID and opens the next map after MID in the same users catalog. If no other map is found, then FALSE is returned. Remember that opening a map does not imply that any of the files containing map array data are opened or read.

SUBROUTINE LUMIO(LUN1,LUN2)

LUMIO may be used to specify a range of logical unit numbers that can be used by the MIO package. One logical unit number is required for each map that is open at any given time. LUMIO must be called before any other MIO subroutines. Subsequent calls to LUMIO have no effect.

SUBROUTINE GLMIO(LUN)
SUBROUTINE RLMIO(LUN)

If a logical unit number is required for non-MIO i/o operations, then the subroutine GLMIO may be used to get the value of a logical unit number from the pool that is managed by MIO. After the application program is finished with the logical unit number, it may be released for further use by RLMIO.

SUBROUTINE IDMIO(USER)

IDMIO may be called at any time to specify a default user number. If OPMIO does not specify a user explicitly in the NAME string, then the default is used.

SUBROUTINE DEMIO(MID)

Delete the map (and its disk files) and remove it from the users catalog.

SUBROUTINE UERMIO(ICODE)
SUBROUTINE ERMIO(ICODE)

The subroutine UERMIO is called by the MIO subroutines if an error occurs. ICODE is a two byte integer. The least significant byte specifies what type of an error occurred, RSX/11 I/O error codes are used for system errors. The most significant byte specifies where and in which MIO routine, the error was detected. The MIO subroutines call UERMIO before returning to the calling subroutine.

UERMIO is not supplied with the MIO package, it must be supplied by the application program. A subroutine called ERMIO is supplied which causes execution of the task to cease, and the value of ICODE to be printed. A simple version of UERMIO can be written which calls ERMIO. Alternatively a version of UERMIO may be written to perform any desired error handling. For example, the error code may be stored in a common block variable to be tested after the MIO routine returns.

THE MIO SUBROUTINES:

All the MIO subroutines are callable by DEC FORTRAN 77. The subroutines that interface to MIO depart from ANSI Fortran 77 in that certain subroutine arguments are optional. All text strings are byte arrays of ASCII characters terminated by a zero. For text strings representing map names, MODE and FMT arguments, no distinction is made between upper and lower case letters.

LOGICAL FUNCTION OPMIO(MID,NAME,MODE,LUN)

MID — A workspace array that is used to identify the map. Its size will be announced later but it will be between 700 and 800 bytes.

NAME — A string that identifies the map. The format of the string is open for discussion. See Barry's notes for the use of wildcards in names.

MODE — A string that is one of the following:

'OLD'	The map must already exist.
'NEW'	The map is to be created.
'READ'	The map must already exist, but it may not be modified.
'SCRATCH'	The map is new, but is not to be kept after task termination

LUN — A logical unit number that is to be used by MIO for performing disk i/o operations. If this argument is omitted, MIO will pick a logical unit number for itself.

OPMIO is used to set up the map identifier MID. MID must only be used in calls to other MIO routines. OPMIO must not be called with the MID of a map that has not been closed. Any map may be read or written at any time, except for maps that have been opened in 'READ' mode. An error codition may occur if an attempt is made to read unwritten data.

OPMIO returns FALSE only if the map is OLD or READ and it does not exist.

There is an alternate specification for a map name that bypasses a catalog search via a catalog record number in order to quickly open a map that has been opened at an earlier time.

SUBROUTINE CLMIO(MID)

This subroutine closes the map (and any files) associated with MID. The MID is now released to be used for other purposes. If the map has been opened as 'scratch', any files associated with it will be deleted. If the map is a new one, cataloguing will be complete.

SUBROUTINE RDMIO(MID,ARRAY,NUM,ICELL,ILINE,ICHAN,ISIZE,FMT)

This subroutine reads data from the map associated with MID. A total of