MDMLIB User's Guide

March, 1989

This manual contains information about MDMLIB, a set of miscellaneous utility routines. The library consists of routines for simple file I/O, interfacing to the $PARSE, $ASCTOID, and $TRNLNM system services, manipulating disk quota entries, and performing character conversions on strings.

Revision/Update Information: This manual replaces the *MDMLIB User's Guide* edition dated 28 July 1988.

Engineering Computing Services
Rensselaer Polytechnic Institute
Troy, New York

10 March 1989

Copyright ©1989 Rensselaer Polytechnic Institute. ALL RIGHTS RESERVED.

Permission is given to redistribute this software for non-commercial purposes provided that there is no charge for the distribution beyond handling and materials costs.

Disclaimer: This software is provided ‘‘as is’’. Neither the author nor Rensselaer Polytechnic Institute makes any guarantee as to the suitability, usefulness, performance, or security of the software.

Contents

PREFACE v

NEW FEATURES vii

CHAPTER 1 INTRODUCTION 1–1

1.1 USING MDMLIB 1–1

1.2 CONTACTING THE AUTHOR 1–1

Part I ROUTINE DESCRIPTIONS

MDM_ASCTOID RTN–3
MDM_CLOSEF RTN–5
MDM_DQ_ADD RTN–6
MDM_DQ_CLOSE RTN–8
MDM_DQ_OPEN RTN–9
MDM_DQ_READ RTN–10
MDM_DQ_REMOVE RTN–12
MDM_DQ_SET RTN–13
MDM_MKDATE RTN–15
MDM_OPENF RTN–17
MDM_PARSE RTN–19
MDM_READF RTN–21
MDM_RWNDF RTN–22
MDM_STRANS RTN–23
MDM_SYS_TRNLNM RTN–25
MDM_WRITEF RTN–27

Preface

MDMLIB is a library of routines used by various ECS utilities. They were written mainly to simplify access to existing VMS system services.

The routines can be grouped as follows:

Sequential File I/O Routines	
MDM_OPENF	Open a file
MDM_READF	Read a record from file
MDM_WRITEF	Write a record to file
MDM_RWNDF	Rewind to beginning of file
MDM_CLOSEF	Clsoe file
Disk Quota Routines	
MDM_DQ_OPEN	Begin disk quota operations
MDM_DQ_ADD	Add quota entry to quota file
MDM_DQ_READ	Read one or more quota entries
MDM_DQ_SET	Modify an existing quota entry
MDM_DQ_REMOVE	Remove a quota entry
MDM_DQ_CLOSE	End disk quota operations
Miscellaneous Routines	
MDM_ASCTOID	Convert UIC or identifier string to numeric value
MDM_MKDATE	Create an RFC822-compliant date/time string
MDM_PARSE	Simplified $PARSE interface
MDM_STRANS	ASCII/EBCDIC/negative-ASCII character translation
MDM_SYS_TRNLNM	Logical name translation á la LIB$SYS_TRNLOG

New Features

All MDMLIB routines have been rewritten in BLISS. New routines added since the last release include disk quota manipulation routines and a routine to convert an character string containing an identifier or UIC specification into the appropriate numeric value.

1 Introduction

MDMLIB is a set of miscellaneous routines used in various ECS projects. The library includes a set of RMS interface routines for using simple sequential files, a routine to make date/time strings that are compliant with RFC 822, routines that interface with the $TRNLNM, $ASCTOID, and $PARSE system services, a routine for performing character conversions, and routines for manipulating disk quotas.

Most of these routines are very similar to routines that already exist on VAX/VMS, or perform tasks that can be accomplished through high-level language constructs (such as the file I/O routines). The MDMLIB routines were developed generally because the extant routines and I/O constructs did not provide exactly the services required.

1.1 Using MDMLIB

The MDMLIB routines be callable from any VAX language; just link with library MDMLIB.OLB. A library of PL/I inserts (PLIMDM.TLB) and a BLISS REQUIRE file (BLISSMDM.R32) accompany the library for use of the routines from those languages. The PL/I text library also includes some preprocessor declarations and procedures to assist in using VMS system services.

1.2 Contacting the Author

Comments, suggestions, and questions about MDMLIB can be directed to the author via electronic mail at one of the following addresses:

Internet: madison@vms.ecs.rpi.edu
BITNET: madison@rpiecs

Part I Routine Descriptions

This part describes the each of the MDMLIB routines.

MDM_ASCTOID—Translate Identifier Name or UIC to Identifier

MDM_ASCTOID translates an ASCII identifier or UIC specification into an identifier value.

FORMAT

MDM_ASCTOID *ascid, ident*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

ascid

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**
ASCII representation of UIC or identifier.

ident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**
Identifier value of UIC or identifier name.

DESCRIPTION

This routine uses LIB$TPARSE and $ASCTOID to parse and translate an identifier or UIC specification from ASCII into the appropriate identifier value.

A UIC specification or identifier may take any of the following forms:

[g,m]	Group and member specified as octal numbers
[alphag,m]	Group as alphanumeric identifier; member in octal
[alphag,alpham]	Group and member as alphanumeric identifiers
[alpham]	Alphnumeric identifier for full UIC
[octal,*]	Wildcard for entire group
[alphag,*]	Wildcard for entire group
[*,*]	Wildcard for all users in all groups
alphaid	Plain, ordinary alphanumeric identifier
%Xhexnum	Identifier value in hexadecimal

These are the forms generally recognized by DCL and its utilities.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from LIB$TPARSE, LIB$ANALYZE_SDESC, and $ASCTOID are possible.

MDM_CLOSEF—Close a file

MDM_CLOSEF closes a file that was opened with MDM_OPENF.

FORMAT

MDM_CLOSEF *unit*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

handle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Handle of open file, as returned by MDM_OPENF.

DESCRIPTION

This routine closes a file that was opened with MDM_OPENF.

Warning about **handle**: Handles are for internal use by MDMLIB routines only, and should not be manipulated by a user program.

CONDITION VALUES RETURNED

RMS$_NORMAL	indicates normal successful completion.

MDM_DQ_ADD—Add a disk quota entry

MDM_DQ_ADD adds a disk quota entry to a disk volume.

FORMAT

MDM_DQ_ADD *dqhandle, ident, permq, overdr*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Handle created with MDM_DQ_OPEN.

ident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
UIC code or identifier value for which the disk quota entry is to be added.

permq

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Permanent disk quota value to be entered.

overdr

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Overdraft disk quota value to be entered.

DESCRIPTION

This routine adds a new entry in the disk quota file for the specified UIC or identifier value.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status value from $QIO or I/O status returns from the ACP-QIO interface driver are possible.

MDM_DQ_CLOSE—Close disk channel

MDM_DQ_CLOSE closes a channel opened with MDM_DQ_OPEN.

FORMAT

MDM_DQ_CLOSE *dqhandle*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**

Handle created by MDM_DQ_OPEN identifying channel to be closed.

DESCRIPTION

MDM_DQ_CLOSE closes the channel opened with MDM_DQ_OPEN and deallocates the memory used for the disk quota structures.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion.

MDM_DQ_OPEN—Open channel to disk

MDM_DQ_OPEN opens a channel to a disk volume for disk quota operations.

FORMAT

MDM_DQ_OPEN *devnam, dqhandle*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

devnam

VMS Usage: **device_name**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Device name of disk device to be opened.

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Handle used to refer to disk for other MDM_DQ operations.

DESCRIPTION

MDM_DQ_OPEN assigns a channel to the specified disk voolume and allocates the structures needed for subsequent disk quota operations. Note that this routine does not attempt to determine whether disk quotas are enabled on the volume.

Note that you must be suitably privileged to perform disk quota manipulations on a volume.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from LIB$GET_VM and $ASSIGN are possible.

MDM_DQ_READ—Read a disk quota entry

MDM_DQ_READ reads a disk quota and usage entry from the quota file.

FORMAT

MDM_DQ_READ *dqhandle, ident, permq, overdr, usage [,context, actident]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Handle for disk channel, as returned by MDM_DQ_OPEN.

ident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

UIC or identifier value for the entry to be read.

permq

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Permanent quota value from disk quota entry.

overdr

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Overdraft value from disk quota entry.

usage

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Disk usage value from disk quota entry.

context

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **modify**
mechanism: **by reference**
Context information for wildcard searches of quota file.

actident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
UIC or identifier value read from disk quota entry during wildcard search.

DESCRIPTION

This routine reads the current quota, overdraft, and usage values for the specified UIC or identifier value. With the use of the **context** and **actident** arguments, this routine can be called repeatedly for a wildcard search of the quota file.

Two types of wildcard searching can be performed. The entries for all users in a particular UIC group can be obtained by specifying UIC$K_WILD_MEMBER for the member portion of the UIC. The entries for all users in all groups can be obtained by specifying UIC$K_WILD_GROUP for the group portion of the UIC and UIC$K_WILD_MEMBER for the member portion of the UIC.

To perform a wildcard search, initialize the **context** variable to zero before the first call to MDM_DQ_READ. The actual UIC or identifier read from the quota file is returned in **actident**. Do not alter the **context** value after the initial read, and continue reading until an error code is returned indicating no further entries.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion.
SS$_BADPARAM	Wildcard UIC specified without **context** and **actident** arguments. Other status values from $QIO and the QIO-ACP driver are also possible.

MDM_DQ_REMOVE—Remove a disk quota entry

MDM_DQ_REMOVE removes a disk quota entry from the quota file.

FORMAT

MDM_DQ_REMOVE *dqhandle, ident*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Handle created with MDM_DQ_OPEN.

ident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

UIC code or identifier value for which the disk quota entry is to be removed.

DESCRIPTION

This routine removes disk quota entries. Note that the entry is removed regardless of whether the usage for the specified identifier is zero or not.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from $QIO or the QIO-ACP driver are possible.

MDM_DQ_SET—Modify a disk quota entry

MDM_DQ_SET sets a new quota and/or overdraft value for one UIC or identifier in the quota file.

FORMAT

MDM_DQ_SET *dqhandle, ident, [permq], [overdr]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

dqhandle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Handle created with MDM_DQ_OPEN.

ident

VMS Usage: **uic_code**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
UIC code or identifier value for which the disk quota entry is to be modified.

permq

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Permanent disk quota value to be entered. If omitted, the permanent quota remains unchanged.

overdr

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Overdraft disk quota value to be entered. If omitted, the overdraft value remains unchanged.

MDM_DQ_SET

DESCRIPTION

This routine modifies a disk quota entry for a specific UIC or identifier value. You can change either the permanent quota or overdraft value or both; if you omit a value it remains unchanged. Use MDM_DQ_REMOVE to remove a disk quota entry from the quota file.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from $QIO or the ACP-QIO driver are possible.

MDM_MKDATE—make RFC822 date string

MDM_MKDATE constructs an RFC822-compliant date/time string from a VMS date/time stamp.

FORMAT

MDM_MKDATE *[fs-date], fmt-date [,fd-len]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

fs-date

VMS Usage: **date_time**
type: **quadword (unsigned)**
access: **read only**
mechanism: **by reference**
A VMS absolute date/time stamp. If omitted, the current system date/time is used.

fmt-date

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
Character string into which the formatted date is written.

fd-len

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
Length of character string stored in **fmt-date**.

DESCRIPTION

This routine constructs a date/time string that complies with the Internet RFC822 date/time specification. The format of the string is:

Dow, *dd Mmm yy hh*:*mm*:*ss zone*

Where *Dow* is a three-letter abbreviation of the day of the week, *dd* is the day of the month, *Mmm* is a three-letter abbreviation for the name of the month, *yy* is the year (last two digits only), *hh*:*mm*:*ss* is the time of day (using 24-hour notation), and *zone* is the time-zone specification.

Note: this routine assumes operation in the Eastern time zone under the current daylight savings time schedule.

CONDITION VALUES RETURNED

SS$_NORMAL	indicates normal successful completion. Status values from LIB$DAY_OF_WEEK and LIB$SYS_FAO are also possible.

MDM_OPENF—Open a file

MDM_OPENF opens a file for sequential access.

FORMAT

MDM_OPENF *key, fspec [,handle] [,defspec] [,resspec]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

key

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Function code indicating operation to be performed. **Key** may have one of the following values:

Value	Meaning
MDM__EXISTS	Test only for file's existence.
MDM__READ	Open file for reading.
MDM__WRITE	Open file for writing.
MDM__APPEND	Open file for appending.

fspec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification of file to be opened.

handle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

''Handle'' assigned to opened file, used to identify the file for subsequent file operations. This argument is optional only when **key** has the value MDM_ _EXISTS.

defspec

VMS Usage: **char_string**
type: **character string**
access: **read only**

mechanism: **by descriptor**
Default file specification to be used when opening the file. If omitteed, a default file specification of .DAT is used.

resspec

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
If specified, the RMS resultant file specification is copied to this string.

DESCRIPTION

MDM_OPENF provides simplified access to RMS for use with sequential disk files. Certain assumptions are made by this routine:

- ''Opening a file for writing'' means creating a new file that is opened for writing. ''Opening a file for appending'' means opening an existing file for writing and positioning to the end of file before the first write operation. Both of these types of open operations use exclusive access to the file.
- Read-only operations allow for multiple readers.
- Files opened for writing have variable-length records and carriage-return carriage control (i.e. typical VMS sequential text file).

The **handle** is actually the address of a control structure used internally by the file I/O routines that should not be manipulated by user programs.

CONDITION VALUES RETURNED

RMS$_NORMAL	Normal successful completion.
MDM__INVFUNC	Invalid value for **key**.
MDM__INVFSPEC	Invalid file specification given for **fspec**, **defspec**, or **resspec**. Error codes from $OPEN, $CREATE, $CONNECT, LIB$GET_VM, and LIB$ANALYZE_SDESC are also possible.

MDM_PARSE—Interface to $PARSE

MDM_PARSE is a simplified interface to the RMS $PARSE service.

FORMAT

MDM_PARSE *keymask, fspec, [defspec,] result [,reslen]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

keymask

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A mask indicating actions to be performed. Any combination of the following values may be used:

Value	Meaning
MDM_M_NODE	Return node specification.
MDM_M_DEV	Return device specification.
MDM_M_DIR	Return directory specification.
MDM_M_NAME	Return file name.
MDM_M_TYPE	Return file type.
MDM_M_VER	Return version number.
MDM_M_ALL	Return all parts of file specification.
MDM_M_SYNCHK	Perform syntax check only.
MDM_M_PWD	Leave password in DECnet access string unaltered.
MDM_M_NOCONCEAL	Fully expand concealed device and root specifications.

fspec

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

File specification to be parsed.

defspec

VMS Usage: **char_string**
type: **character string**
access: **read only**

mechanism: **by descriptor**
Default file specification to be used while parsing.

result

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
Result of the parsing operation. The result includes only those parts of the file specification specified by **keymask**.

reslen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
Length of string written to **result**.

DESCRIPTION

This routine provides a fairly simple interface to the $PARSE RMS service. Specify which parts of the file specification you want returned, and those parts, written in normal RMS order with appropriate punctuation, are returned in **result**. The SYNCHK, PWD, and NOCONCEAL options are passed along to $PARSE. Refer to the RMS manual for more information.

CONDITION VALUES RETURNED

RMS$_NORMAL	Normal successful completion. Other status values from $PARSE, LIB$GET_VM, LIB$SCOPY_DXDX, and LIB$ANALYZE_SDESC are possible.

MDM_READF—Read a record

MDM_READF reads a record from a file opened by MDM_OPENF.

FORMAT

MDM_READF *handle, recbuf [,len]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

handle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Handle of open file, as returned by MDM_OPENF.

recbuf

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
Character string into which the record read from the file is stored.

len

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
Length of record read from file.

DESCRIPTION

MDM_READF reads in one record from a file opened by MDM_OPENF. The RTL character string routines are used to copy the record into **recbuf**, so any type of character string descriptor that the RTL handles can be used.

CONDITION VALUES RETURNED

RMS$_NORMAL	Normal successful completion.
RMS$_EOF	Attempt was made to read beyond end of file. Other status values from $GET are also possible.

MDM_RWNDF—Rewind file

MDM_RWNDF rewinds a file that was opened with MDM_OPENF.

FORMAT

MDM_RWNDF *handle*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

handle
VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

DESCRIPTION

This routine simply issues a $REWIND RMS service on the file.

CONDITION VALUES RETURNED

RMS$_NORMAL	Normal successful completion. Other status values from $REWIND are also possible.

MDM_STRANS—Interface to MOVTC instruction

MDM_STRANS translates character strings using the MOVTC instruction.

FORMAT

MDM_STRANS *func, srcstr, dststr*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

func

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Selects the translation function to be performed. **Func** can have one of the following values:

Value	Meaning
MDM__ASCEBC	ASCII to EBCDIC translation.
MDM__EBCASC	EBCDIC to ASCII translation.
MDM__ASCPRI	ASCII to Prime-ASCII translation.
MDM__PRIASC	Prime-ASCII to ASCII translation.

srcstr

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

String to be translated.

dststr

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

Result of translation.

DESCRIPTION

This routine translates between ASCII and EBCDIC as well as between ASCII and ''Prime ASCII'' (also called ''negative ASCII''), which is 7-bit ASCII codes with the 8th bit set in each byte.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from STR$GET1_DX, LIB$ANALYZE_SDESC, and LIB$SCOPY_DXDX are possible.

MDM_SYS_TRNLNM—Interface to $TRNLNM service

MDM_SYS_TRNLNM translates logical names using $TRNLNM.

FORMAT

MDM_SYS_TRNLNM *lognam, [attr], [tblmask], [acmode], eqvnam [,eqvlen]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

lognam

VMS Usage: **logical_name**
type: **character string**
access: **read only**
mechanism: **by descriptor**
Logical name to be translated.

attr

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Attributes controlling the logical name search, as used by $TRNLNM.

tblmask

VMS Usage: **mask_longword**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Mask specifying which tables should be searched.

Bit number	Mask value	Meaning
3	8	Process logical name table
2	4	Job logical name table
1	2	Group logical name table
0	1	System logical name table

acmode

VMS Usage: **access_mode**
type: **byte (unsigned)**
access: **read only**

mechanism: **by reference**
Access mode to be used in the translation, as used by $TRNLNM.

eqvnam

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
String into which the equivalence name is written if found.

eqvlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
Length of equivalence name.

DESCRIPTION

This routine uses the $TRNLNM system service to translate a logical name. Only the tables specified by the **tblmask** argument are searched. The tables are searched in the following order: process, job, group, system. Only the first equivalence name can be retrieved with this routine–search lists cannot.

See the description of the $TRNLNM system service for an explanation of the **attr** and **acmode** arguments.

CONDITION VALUES RETURNED

SS$_NORMAL	Normal successful completion. Other status values from LIB$SCOPY_DXDX, LIB$SCOPY_R_DX, and $TRNLNM are possible.

MDM_WRITEF—Write a record to a file

MDM_WRITEF writes a record to a file opened by MDM_OPENF.

FORMAT

MDM_WRITEF *handle, recbuf*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

handle

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Handle of file, as returned by MDM_OPENF.

recbuf

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**
Character string containing data to be written to the file.

DESCRIPTION

This routine uses the $PUT RMS service to write the record to the file. The file must have been opened for writing (or appending) with MDM_OPENF.

CONDITION VALUES RETURNED

RMS$_NORMAL	Normal successful completion. Other status values from LIB$ANALYZE_SDESC and $PUT are possible.