3.07.001

THIS MANUAL IS CURRENT FOR MDBS 3.07

MDBS III

System Specific Installation Manual

for

CP/M with PL/I-80

Micro Data Base Systems, Inc.

P.O. Box 248

Lafayette, Indiana 47902

USA

November 1981
Revised October 1984

NEW RELEASES, VERSIONS, AND A WARNING

Any programming endeavor of the magnitude of the MDBS software will necessarily continue to evolve over time. Realizing this, Micro Data Base Systems, Inc., vows to provide its users with updates to **this version** for a nominal handling fee.

New versions of MDBS software will be considered as separate products. However, bona fide owners of previous versions are generally entitled to a preferential rate structure.

Finally, each copy of our software is personalized to identify the licensee. There are several levels of this personalization, some of which involve encryption methods guaranteed to be combinatorially difficult to decipher. Our products have been produced with a very substantial investment of capital and labor, to say nothing of the years of prior involvement in the data base management area by our principals. Accordingly, we are seriously concerned about any unauthorized copying of our products and will take any and all available legal action against illegal copying or distribution of our products.

DISCLAIMER NOTICE

Although care has been taken in the preparation of this material, Micro Data Base Systems, Inc. assumes neither responsibility for errors or omissions, nor for damages resulting from the use of the information contained herein. Micro Data Base Systems, Inc. does not warrant its accuracy nor guarantee the operation of the system in every instance described herein.

The reader/user assumes full liability and all risk for any damages resulting from the use of the information contained herein, and for determining whether the information contained herein is suitable for user's intended purpose.

Micro Data Base Systems, Inc. reserves the right to incorporate design improvements and new functions in its software products and software systems. Recent improvements may not always be reflected in documentation.

PREFACE

Although the great majority of MDBS features and facilities are independent of the host operating system and host programming languages, there are some system specific aspects. These include the installation procedures, execution command lines, DML command forms, and data item-host language variable correspondences. This manual presents the system specific aspects that are needed in order to use MDBS DDL/DMS, MDBS-QRS, MDBS-RCV, MDBS-DMU and MDBS-IDML.

This manual consists of the following eight chapters:

I. FILE NAMES

- A. File Names for MDBS Software
- B. Fully Qualified File Names in CP/M
- C. Special Keys when Using Interactive MDBS Software under CP/M
- D. MP/M Environments
- E. Contention Count Time
- F. The Renaming Utility

II. INSTALLATION and TESTING PROCEDURES

- A. Installing MDBS.DDL and MDBS.DMS
- B. Testing

III. INVOKING MDBS.DDL

IV. OPERATING SYSTEM DEPENDENT DEFAULTS

- A. File name defaults for areas
- B. File extension defaults
- C. Pages per area default
- D. Page size default
- E. Page size restrictions

V. DATA ITEM - HOST LANGUAGE VARIABLE CORRESPONDENCE

- A. Non-numeric Data Items
- B. Integer Data Items
- C. Unsigned Data Items
- D. Internal Decimal Data Items
- E. Real Data Items
- F. Repeating Data Items

VI. CONTROL PROCEDURES

- A. Running an Application Program
- B. Special Link Files

I. FILE NAMES

A. File Names for MDBS Software

The MDBS software and the software of MDBS add-on packages are furnished in a collection of files. In the CP/M:PL/I (2.2 and later versions of CP/M) environment, these files have the following names:

READ.ME	If this file is provided, it contains special notes, announcements and comments which you should read.
FNDMS.PLI	external definition include file
DMS.REL	MDBS.DMS library
PLI80.REL	PL/I language interface
A.REL	MDBS.DMS component
OS.REL	CP/M interface
Z.REL	MDBS.DMS component
RTL.REL	MDBS.RTL library (used in place of DMS.REL)
DATACOD.COM	DATACOD relocatable file utility
DDL.COM	MDBS.DDL object code
DDL1.OVL	MDBS.DDL overlay 1
DDL2.OVL	MDBS.DDL overlay 2
DDL3.OVL	MDBS.DDL overlay 3
SAMPLE.PLI	direct call sample program (for use with SAMPLE.DDL)
SAMPLE.DDL	sample ddl specification
QRS.COM	MDBS.QRS object code
QRS1.OVL	QRS support overlay 1
QRS2.OVL	QRS support overlay 2
QRS3.OVL	QRS support overlay 3
QRS4.OVL	QRS support overlay 4
QRS5.OVL	QRS support overlay 5
QRS6.OVL	QRS support overlay 6
QRS7.OVL	QRS support overlay 7
QRS8.OVL	QRS support overlay 8
IDML.COM	MDBS.IDML object code
IDML1.OVL	IDML support overlay 1
IDML2.OVL	IDML support overlay 2
IDML3.OVL	IDML support overlay 3
IDML4.OVL	IDML support overlay 4
IDML5.OVL	IDML support overlay 5
RIDML.COM	MDBS.IDML object code (RTL form)
RIDML1.OVL	RIDML support overlay 1
RIDML2.OVL	RIDML support overlay 2
RIDML3.OVL	RIDML support overlay 3
RIDML4.OVL	RIDML support overlay 4
RIDML5.OVL	RIDML support overlay 5
UTIL.ERR	error messages for QRS, IDML
DMS.ERR	error messages for DMS
DMU.COM	MDBS.DMU object code
RCV.COM	object code for the MDBS-RTL recovery program:RCV
RCV1.OVL	RCV support overlay 1
RCV2.OVL	RCV support overlay 2
CNV.COM	object code for the DDL conversion program:CNV

C. Special Keys when Using Interactive MDBS Software under CP/M

RETURN (ENTER)	terminates an input line
CONTROL-X	interrupts a line entry and restarts the input line
CONTROL-H	causes a character deletion in the line being input
CONTROL-I	causes a tab character to be placed in the line
CONTROL-C	returns control to the operating system (hard interrupt)
ESCAPE	causes the prompt of the interactive software to appear (soft interrupt)
CONTROL-P	toggles the interactive software output between the console and printer
CONTROL-S	causes a pause in the output from interactive software
CONTROL-Q	causes output from interactive software to resume, following a CONTROL-S pause

D. MP/M Environments

All MDBS software for use under CP/M (versions 2.2 and later) can also be used under MP/M. This manual applies equally to MP/M and CP/M. **Note:** MP/M is too large to allow the use of MDBS-IDML or MDBS-QRS with a one user configuration. They can be used under MP/M with the 1-4, and over 4 multiuser versions of MDBS.

MDBS access speeds depend on many factors including the extent of an application, the quality of scheme design, the host language used, the quality of application programming, data volume, the hardware used, and the operating system. Due to the directory utilization approaches of CP/M and MP/M, it is generally true that MDBS provides faster access under CP/M than under MP/M.

E. Contention Count Time

The unit of time used with MP/M for the DMS contention count command is one clock tick (i.e., 1/50 or 1/60 of a second). See the MCC command in the **MDBS DMS Manual.**

II. INSTALLATION AND TESTING PROCEDURES

A. Installation

1. MDBS.DDL

MDBS.DDL is installed by simply copying ("PIP"ing with the ov option) the DDL.COM, DDL1.OVL, DDL2.OVL, and DDL3.OVL files to a working disk. Because MDBS.DDL uses an overlay technique, this working disk must reside on the default drive in order to execute.

2. MDBS.DMS

Prior to linking MDBS.DMS and your application program, you need to create a CP/M DMS library (to be called CPMDMS.IRL). This requires the use of CP/M's library manager, LIB. Create CPMDMS.IRL by entering the following operating system command line:

```
LIB CPMDMS[I] = DMS,OS,A,Z
```

If RTL is to be used, the installation procedure is the same, except that you should enter CPMRTL instead of CPMDMS and RTL instead of DMS; this creates a CP/M RTL library. The library creation process assumes that the DMS.REL (or RTL.REL for RTL), OSCPM.REL, A.REL, and Z.REL files are on the default drive.

Now copy CPMDMS.IRL (or CPMRTL.IRL for RTL), PLI80.REL, and FNDMS.PLI to a working disk.

III. INVOKING MDBS.DDL

The operating system command string for executing MDBS.DDL is:

```
DDL fully-qualified-file-name  -Bnnnn
```

where the fully-qualified-file-name and -Bnnnn arguments are optional. If the fully-qualified-file-name is omitted, then the MDBS.DDL program responds with the :: prompt and is ready for interactive usage (see VI-A,B of the MDBS DDL Manual). If a fully-qualified-file-name is specified, then MDBS.DDL is executed on a batch basis (see VI-C of the MDBS DDL Manual). The contents of this file must be a valid DDL specification. For instance,

```
DDL TRIAL.DDL
```

will cause MDBS.DDL to analyze the DDL specification contained in the TRIAL.DDL file on the default drive.

The other optional argument (-Bnnnn) can be ignored in this environment.

V. DATA ITEM - HOST LANGUAGE VARIABLE CORRESPONDENCE

This chapter shows the type, size, and value correspondences that exist between MDBS data items and PL/I variables. Correct usage of DML create, put, and get commands depends on a knowledge of these correspondences. Other DML commands (e.g., FMSK) also require input from a PL/I variable, where the variable must be consistent with a data item of a particular type and size.

A. Non-numeric Data Items

MDBS Data Item		PL/I Variable	
MDBS type	MDBS size	PL/I type	PL/I size
binary	n	character	n
character	n	character	n
string	n	character	n varying
date	-	character	10
time	-	character	9

Since the maximum size for a PL/I character variable is 254, no more than 254 bytes can be stored from a PL/I variable into a data item whose type is binary or character. If the size of an MDBS binary or character data item exceeds 254, then right blank fill occurs during storage and right truncation occurs during retrieval.

B. Integer Data Items

The host language variables that are consistent with various sizes of an integer data item are presented in Table V-1. This table also shows the mappings of data values from PL/I variables into integer data items during data storage (e.g., CRS, PFM, etc.). Similarly, the mappings of data values from integer data items to corresponding PL/I variables during data retrieval (e.g., GFM) are shown.

As an example, when storing a data value from a fixed binary (7) variable into a one byte integer data item, the value must be in the range -128 to 127. Any other value for the fixed binary (7) variable will not be permitted and the DML command that attempts to store such a value will return a command status of 33. When retrieving a data value from a 3 byte integer data item into a fixed binary (15)

variable, an appropriate value in the range -32768 to 32767 is deposited in the fixed binary (15) variable. If the stored value is outside of this range, then the value of the PL/I variable is undefined. As a third example, suppose we want to store the value -32700 into a two byte integer data item. This is accomplished with a put command that uses a fixed binary (15) variable having the value -32700.

C. Unsigned Data Items

The host language variables that are consistent with various sizes of an unsigned data item are presented in Table V-2. This table also shows the mappings of data values from PL/I variables into unsigned data items during data storage (e.g., CRS, PUTM, etc.). Similarly, the mappings of data values from unsigned data items to corresponding PL/I variables during data retrieval (e.g., GETM) are shown.

As an example, when storing a data value from a fixed binary (7) variable into a one byte unsigned data item, the variable's value must be in the range -128 to 127.

When retrieving a data value from a 3 byte unsigned data item into a fixed binary (15) variable, an appropriate value in the range -32768 to 32767 is deposited into the fixed binary (15) variable. If the unsigned stored value is 65533, then it is returned as the value -3. If the unsigned value is greater than 65535, then the value of the PL/I variable will be undefined.

As a third example, suppose we want to store the value 32769 into a two byte unsigned data item. This is accomplished with a put command that uses a fixed binary (15) variable having the value -32767.

D. Internal Decimal Data Items

The host language variables that are consistent with various sizes of an idec data item are presented in Table V-3. This table also shows the largest relative error that can occur when storing data into various sizes of idec data items and when retrieving data from various sizes of idec data items.

When storing data values into an idec data item, there is no potential for overflow. When retrieving data from an idec data item whose size does not exceed fifteen digits (i.e., $n \leq 15$), overflow occurs if the stored data value has an absolute value greater than $(10^n-1)/10^d$. When retrieving data from an idec data item whose size exceeds fifteen digits (i.e., $n > 15$), overflow occurs if the stored data value has an absolute value greater than $(10^{15}-1)/10^d$.

Table V-3. Internal Decimal Correspondences

MDBS Data Item		PL/I Variable	Storing Data	Retrieving Data
MDBS type	MDBS size	PL/I type	Largest Relative Error	Largest Relative Error
idec n,d	n=1,2...or 15	fixed decimal(n,d)	0	0
idec n,d	$n \geq 16$	fixed decimal(15,d)	0	$5.000 * 10^{-16}$

E. Real Data Items

The host language variables that are consistent with various sizes of a real data item are presented in Table V-4. This table shows the largest relative error that can occur when storing data into various sizes of real data items and when retrieving data from various sizes of real data items.

Table IV also shows the overflow potential when storing and retrieving data. For instance, an attempt to store $1.7014 * 10^{38}$ into a three byte real data item will result in an undefined value for that data item value.

F. Repeating Data Items

When storing data into or retrieving data from a repeating data item, a PL/I array is used. The appropriate kind of array for each data item type and size is shown below, where m represents the number of replications defined for the data item in the DDL specification (with an occurs clause). Here, "data" is the host language array being used for storage or retrieval.

Repeating Data Item (m replications)		Form of the PL/I Variable Array: data
MDBS type	MDBS size	
binary	n	data(m) char(n)
character	n	data(m) char(n)
string	n	data(m) char(n) varying
date	-	data(m) char(10)
time	-	data(m) char(9)
integer	1	data(m) fixed(7)
integer	≥ 2	data(m) fixed
unsigned	1	data(m) fixed(7)
unsigned	≥ 2	data(m) fixed
idec	n=1,2,...,or 15	data(m) fixed decimal(n,d)
idec	≥ 16	data(m) fixed decimal(15,d)
real	n	data(m) float

VI. CONTROL PROCEDURES

A. Running an Application Program

The following steps are used to control the selective interfacing of MDBS.DMS routines with a PL/I application program. They assume that installation as described in Chapter II has been completed.

1. Create your application program using the direct DML command form (see Chapter VII). All DML commands used in the program must be declared to be externals, either explicitly or using the **%include** command with an edited copy of FNDMS.PLI (containing only those external declarations needed by the program). This is illustrated in Chapter VII.

2. Compile your program in the usual manner. For instance,

   ```
   PLI PRG
   ```

   where PRG is the file containing the PL/I program source code.

3. Selectively link your compiled program and MDBS.DMS together in the following way:

   ```
   LINK PRG[A],PLI80,CPMDMS.IRL[S]
   ```

   where PRG.REL contains the compiled program. This assumes that the program is on the working disk that contains PLI80.REL and CPMDMS.IRL (or CPMRTL.IRL in the case of RTL). It also assumes that this disk is on the default drive.

4. Execute the linked program.

B. Special Link Files

Special link files are provided which can be used to optimize performance in certain situations. These are FASTIO, NOCALC, NOFLOAT, NOREAL, NOIDEC, NODATE, and NOTIME. These are not always available in all environments under all releases of MDBS. Those which are available can be linked with the application program. If FASTIO is desired, you should create CPMDMS with the command line

```
LIB CPMDMS[I]=DMS,FASTIO,OS,A,Z
```

instead of the command line shown in Chapter II. If NOCALC is desired, then it is linked by inserting NOCALC immediately prior to PLI80 in the LINK command line. If you desire to link any of the other special files, then insert the file name(s) prior to PLI80 and use PLI80[S] in place of PLI80 on the LINK command line.

Each of the special link files (except FASTIO) disables certain MDBS features. If you invoke a DML command that attempts to process a disabled feature, then a command status of 34 results.

VII. DML COMMAND FORMAT

PL/I is a record oriented language that permits direct invocation of DML commands (see the record/direct example for each DML command in the MDBS DMS Manual). The precise calling forms for direct DML usage presented in Appendix A and are illustrated in the examples below.

A. Command Status and Required Declarations

The command status variable must be declared to be fixed.

All DML commands used in a PL/I program must be defined as externals. The following declarations are required in a PL/I host language program:

```
dcl
  .
  .
  .
external definitions for all DML commands to be used in program
  .
  .
  .
e0 fixed;
```

The external declarations for the DML commands appear in Appendix B. These declarations are also furnished on the FNDMS.PLI file. This file can be used in conjunction with the PL/I **%include** command. This is an alternative to explicitly specifying external definitions. It is accomplished with the following declarations:

```
%include 'FNDMS.PLI'
dcl
e0 fixed;
```

B. Special Commands

The only special commands that can be used in this environment are SETPBF and ALTEOS. These commands are not required.

D. Find Command Examples

```
%include 'FNDMS.PLI'
dcl
var1 char(5)
var2 fixed binary(7)
 .
 .
 .
e0 = fmsk ('set1',addr(var1));
e0 = fmsk ('set5',addr(var2));
```

Here, set1 and set5 are names of sets that have been specified in a DDL specification.

E. Get and Put Command Examples

```
%include 'FNDMS.PLI'
dcl
var fixed decimal (7,2);
 .
 .
 .
e0 = gfc('ytdearn',addr(var));
var = var + 1029.00
e0 = pfc('ytdearn',addr(var));
```

F. Assignment Command Examples

```
%include 'FNDMS.PLI'
dcl
var fixed;
 .
 .
 .
var = 3;
e0 = smu('set1',addr(var));
 .
 .
 .
e0 = som('set1,set2');
```

Here, set1 and set2 are names of sets that have been specified in a DDL specification.

VIII. INTERACTIVE ADD-ON PACKAGES

MDBS add-on packages are provided on COM files. The packages can be invoked as follows:

A. MDBS-CNV

To invoke the interactive MDBS.CNV program, the following operating system command line is used:

```
CNV
```

B. MDBS-IDML

Before using IDML, be sure that the following files reside on a working disk on the default drive:

IDML.COM, IDML1.OVL, IDML2.OVL, IDML3.OVL, IDML4.OVL, IDML5.OVL

Omitting IDML3.OVL has the effect of disabling the IDML DEFINE command.

To invoke the interactive MDBS.IDML program, the following operating system command line is used:

```
IDML
```

The user can optionally specify the name of an alternative startup file and/or the -B parameter on this command line. The default startup file must have the name: STARTUP. If an alternative file name is used on the command line, it must be fully qualified (see I-B). If the -B parameter is used, it must be followed by the number of bytes being allocated. This number should exceed the minimum DMS buffer region size displayed by MDBS.DDL during data base initialization (VI-B-4 of the MDBS DDL Manual). For example, to use the startup information on the file START.IDM and allocate 2560 bytes, the operating system command line is:

```
IDML START.IDM  -B2560
```

If a DMS command status of 31 results, then the number of bytes should be increased. If an IDML error of insufficient room in memory results with the -B option, then the number of bytes should be reduced.

D. MDBS-DMU

To invoke the interactive MDBS.DMU program, the following operating system command line is used:

```
DMU
```

E. MDBS-RCV

A log file used with the RTL form of MDBS must be a fully qualified file name within CP/M (see I-B). To invoke the interactive MDBS.RCV program, the following operating system command line is used:

```
RCV
```

Be sure to make all necessary backups before using RCV.

The log buffer size in this environment is 128 bytes.

F. MDBS-CBRU

To invoke the Compact-Backup-Restore Utility, the following operating system command is used:

```
CBRU
```


Appendix A

```
e0 = ALTEOS()
e0 = AMM('setl,set2,set3')
e0 = AMO('setl,set2,set3')
e0 = AOM('setl,set2,set3')
e0 = AOO('setl,set2,set3')
e0 = AUI(addr(var))
e0 = CCU(addr(var))
e0 = CRA('record,area',addr(var))
e0 = CRS('record',addr(var))
e0 = DBCLS()
e0 = DBCLSA('area')
e0 = DBENV(addr(var))
e0 = DBOPN(addr(var))
e0 = DBOPNA('area',addr(var))
e0 = DBSAVE()
e0 = DBSTAT(addr(var))
e0 = DRC()
e0 = DRM('set')
e0 = DRO('set')
e0 = FDRK('record',addr(var))
e0 = FFM('set')
e0 = FFO('set')
e0 = FFS(['area'])
e0 = FLM('set')
e0 = FLO('set')
e0 = FMI('item,set',addr(var))
e0 = FMSK('set',addr(var))
e0 = FNM('set')
e0 = FNMI('item,set',addr(var))
e0 = FNMSK('set',addr(var))
e0 = FNO('set')
e0 = FNOI('item,set',addr(var))
e0 = FNOSK('set',addr(var))
e0 = FNS(['area])
e0 = FOI('item,set',addr(var))
e0 = FOSK('set',addr,(var))
e0 = FPM('set')
e0 = FPO('set')
e0 = FRK('record',addr(var))
e0 = GETC(addr(var))
e0 = GETM('set',addr(var))
e0 = GETO('set',addr(var))
e0 = GFC('item',addr(var))
e0 = GFM('item,set',addr(var))
e0 = GFO('item,set',addr(var))
e0 = GMC('set',addr(var))
e0 = GOC('set',addr(var))
e0 = GTC(addr(var))
e0 = GTM('set',addr(var))
e0 = GTO('set',addr(var))
e0 = IMS('set')
e0 = IOS('set')
e0 = LGCPLX()
e0 = LGENDX()
e0 = LGFILE(addr(var))
e0 = LGFLSH()
e0 = LGMSG(addr(var))
e0 = MAU(addr(var))
e0 = MCC(addr(var))
e0 = MCF()
e0 = MCP()
e0 = MRTF(['record'])
e0 = MRTP('record')
e0 = MSF(['set'])
e0 = MSP('set')
e0 = NCI()
e0 = PFC('item',addr(var))
e0 = PFM('item,set',addr(var))
e0 = PFO('item,set',addr(var))
e0 = PIFD(addr(var))
e0 = PUTC(addr(var))
e0 = PUTM('set',addr(var))
e0 = PUTO('set',addr(var))
e0 = RMS('set')
e0 = ROS('set')
e0 = RSM('set')
e0 = RSO('set')
e0 = SCM('set')
e0 = SCN()
e0 = SCO('set')
e0 = SCU(addr(var))
e0 = SETPBF(addr(var),var)
e0 = SMC('set')
e0 = SME('set')
e0 = SMM('setl,set2')
e0 = SMN('set')
e0 = SMO('setl,set2')
e0 = SMU('set',addr(var))
e0 = SOC('set')
e0 = SOE('set')
e0 = SOM('setl,set2')
e0 = SON('set')
e0 = SOO('setl,set2')
e0 = SOU('set',addr(var))
e0 = SUC(addr(var))
e0 = SUM('set',addr(var))
e0 = SUN(addr(var))
e0 = SUO('set',addr(var))
e0 = SUU(addr(var))
e0 = TCN()
e0 = TCT('record')
e0 = TMN('set')
e0 = TMT('record,set')
e0 = TON('set')
e0 = TOT('record,set')
e0 = TRABT()
e0 = TRBGN()
e0 = TRCOM()
e0 = TUN(addr(var))
e0 = XMM('setl,set2,set3')
e0 = XMO('setl,set2,set3')
e0 = XOM('setl,set2,set3')
e0 = XOO('setl,set2,set3')
```

NOTE: [] indicates an optional argument

Appendix B

External Declarations

```
alteos returns (fixed),
amm entry (char(26) var) returns (fixed),
amo entry (char(26) var) returns (fixed),
aom entry (char(26) var) returns (fixed),
aoo entry (char(26) var) returns (fixed),
aui entry (pointer) returns (fixed),
ccu entry (pointer) returns (fixed),
cra entry (char(17), pointer) returns (fixed),
crs entry (char(8), var, pointer) returns (fixed),
dbcls returns (fixed),
dbclsa entry (char(8) var) returns (fixed),
dbenv entry (pointer) returns (fixed),
dbopn entry (pointer) returns (fixed),
dbopna entry (char(8) var, pointer) returns (fixed),
dbsave returns (fixed),
dbstat entry (pointer) returns (fixed),
drc returns (fixed),
drm entry (char(8) var) returns (fixed),
dro entry (char(8) var) returns (fixed),
fdrk entry (char(8) var, pointer) returns (fixed),
ffm entry (char(8) var) returns (fixed),
ffo entry (char(8) var) returns (fixed),
ffs entry (char(8) var) returns (fixed),
flm entry (char(8) var) returns (fixed),
flo entry (char(8) var) returns (fixed),
fmi entry (char(17) var, pointer) returns (fixed),
fmsk entry (char(8) var, pointer) returns (fixed),
fnm entry (char(8) var) returns (fixed),
fnmi entry (char(17) var, pointer) returns (fixed),
fnmsk entry (char(8) var, pointer) returns (fixed),
fno entry (char(8) var) returns (fixed),
fnoi entry (char(17) var, pointer) returns (fixed),
fnosk entry (char(8) var, pointer) returns (fixed),
fns entry (char(8) var) returns (fixed),
foi entry (char(17) var, pointer) returns (fixed),
fosk entry (char(8) var, pointer) returns (fixed),
fpm entry (char(8) var) returns (fixed),
fpo entry (char(8) var) returns (fixed),
frk entry (char(8) var, pointer) returns (fixed),
getc entry (pointer) returns (fixed),
getm entry (char(8) var, pointer) returns (fixed),
geto entry (char(8) var, pointer), returns (fixed),
gfc entry (char(8) var, pointer) returns (fixed),
gfm entry (char(17) var, pointer) returns (fixed),
gfo entry (char(17) var, pointer) returns (fixed),
gmc entry (char(8) var, pointer) returns (fixed),
goc entry (char(8) var, pointer) returns (fixed),
gtc entry (pointer) returns (fixed),
gtm entry (char(8) var, pointer) returns (fixed),
gto entry (char(8) var, pointer) returns (fixed),
ims entry (char(8) var) returns (fixed),
ios entry (char(8) var) returns (fixed),
lgcplx returns (fixed),
lgendx returns (fixed),
lgfile entry (pointer) returns (fixed),
lgflsh returns (fixed),
```

DOCUMENTATION COMMENT FORM

MDBS Document Title:_____

We welcome and appreciate all comments and suggestions that can help us to improve our manuals and products. Use this form to express your views concerning this manual.

Please do not use this form to report system problems or to request materials, etc. System problems should be reported to **MDBS** by phone or telex, or in a separate letter addressed to the attention of the technical support division. Requests for published materials should be addressed to the attention of the marketing division.

Sender:

_____ _____
(name) (position)

_____ _____
(company) (telephone)

_____
(address)

_____
(city, state, zip)

COMMENTS:

Areas of comment are general presentation, format, organization, completeness, clarity, accuracy, etc. If a comment applies to a specific page or pages, please cite the page number(s).

_____

_____

_____

_____

_____

_____

_____

_____

Continue on additional pages, as needed. Thank you for your response.