3.07.001

THIS MANUAL IS CURRENT FOR MDBS 3.07

MDBS III

System Specific Installation Manual

for

CP/M with PASCAL MT+

Micro Data Base Systems, Inc.

P.O. Box 248

Lafayette, Indiana 47902

USA

December 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. MDBS Linker
- C. Testing
- D. Alternative MDBS.DMS Installation Method

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
- C. Alternative Control Procedures

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:PASCAL MT+ (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.PAS	external definition include file
DMS.REL	MDBS.DMS library (standard form)
DBRUN.REL	runtime loader component
FPMT.REL	language interface with floating point real conversion routines
BCDMT.REL	language interface with BCD real conversion routines
A.REL	MDBS.DMS component
OS.REL	CP/M interface
Z.REL	MDBS.DMS component
RTL.REL	MDBS.DMS library (RTL form)
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.PAS	direct call sample program (for use with SAMPLE.DDL)
SAMPLE.DDL	sample ddl specification
QRS.COM	MDBS.QRS object code
QRS0.OVL	QRS support overlay 0
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

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

   If you have PASCAL MT+ Version 5.5 or greater: Use MTPLUS to compile PINI.SRC, thereby producing PINI.ERL.

   If you have version 5.5 or greater (including LIB):
   Prior to linking MDBS.DMS and your application program, you need to create a CP/M DMS library (to be called CPMDMS.REL). This requires the use of a library manager (e.g., Digital Research's LIB). Create CPMDMS.REL by entering the following operating system command line:

   ```
   LIB CPMDMS = DMS,OS,A
   ```

   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), OS.REL, and A.REL files are on the default drive. Now copy CPMDMS.REL (or CPMRTL.REL for RTL), FPMT.REL, BCDMT.REL, and FNDMS.PAS to a working disk.

   If you do not have LIB:
   Copy DMS.REL (or RTL.REL for RTL), OS.REL, A.REL, FPMT.REL, BCDMT.REL, and FNDMS.PAS to a working disk.

B. MDBS Linker

The MDBS linker for selectively linking MDBS.DMS with your application program is on the MLINK.COM, MLINK1.OVL, and MLINK2.OVL files. These files should be on the default drive. MLINK is invoked by:

```
MLINK -Z -X pgm -LCPMDMS -LPASLIB.ERL
```

Here, **pgm** denotes the fully qualified file name (or names), containing compiled program(s) to be linked. If more than one is specified, they should be separated by spaces. The linked program is written to a file having the same name as the first **pgm** file used with MLINK, except it has a .COM extension. An alternative output file name can be specified by including -O**alt** prior to **pgm**, where **alt** is the fully qualified alternative output file name (MLINK -O**alt pgm** ...). The -L option indicates that MLINK will selectively link needed object modules from the indicated REL file (e.g., CPMDMS).

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 PASCAL MT+ 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 PASCAL MT+ 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		
MDBS_type	MDBS_size	PASCAL_MT+_Variable
binary	n	packed array [1:n] of char
character	n	packed array [1:n] of char
string	n	string [n]
date	-	packed array [1:10] of char
time	-	packed array [1:9] of char

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 PASCAL MT+ 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 PASCAL MT+ variables during data retrieval (e.g., GFM) are shown.

As an example, when storing a data value from a PASCAL MT+ integer variable into a one byte integer data item, the value must be in the range -128 to 127. Any other value for the PASCAL MT+ integer 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 three byte integer data item into a PASCAL MT+ integer variable, an appropriate value in the range -32768 to 32767 is deposited in the PASCAL MT+ variable. If the stored value is outside of this range, then the contents of the PASCAL MT+ variable will be undefined. As a third example, suppose we want to store the value 32700 into a four byte integer data item. This is accomplished with a put command that uses a PASCAL MT+ integer 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 PASCAL MT+ 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 PASCAL MT+ variables during data retrieval (e.g., GETM) are shown.

As an example, when storing a data value from a byte variable into a one byte unsigned data item, the variable's value must be in the range 0 to 255.

When retrieving a data value from a 3 byte unsigned data item into a word variable, an appropriate value in the range 0 to 65535 is deposited into the word variable. If the unsigned stored value is 65533, then it is returned as the value 65533. If the unsigned value is greater than 65535, then the value of the PASCAL MT+ variable will be undefined.

D. Internal Decimal Data Items

The host language variables that are consistent with various sizes of an idec data item are presented in Tables V-3a and V-3b. These tables also show 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. Consult Table V-3a when the PASCAL MT+ B option is used. Consult Table V-3b when the B option is not used.

With_the_B_option: When storing data values into an idec data item, there is no potential for overflow. When retrieving data from an idec data item, overflow occurs if the stored data value has an absolute value greater than $1.000 * 10^{14}$.

Without_the_B_option: When storing data values into an idec data item, there is no potential for overflow. When retrieving data from an idec data item, overflow occurs if the stored data value has an absolute value greater than $1.844 * 10^{19}$.

Table V-3a. Internal Decimal Correspondences
(with B option)

MDBS Data Item		PASCAL MT+ Variable	Storing Data	Retrieving Data
MDBS type	MDBS size	PASCAL MT+ type	Largest Relative Error	Largest Absolute Error
idec	1 or 2	real (BCD)	$5.000 * 10^{-3}$	$1.000 * 10^{-4}$
idec	3 or 4	real (BCD)	$5.000 * 10^{-5}$	$1.000 * 10^{-4}$
idec	5 or 6	real (BCD)	$5.000 * 10^{-7}$	$1.000 * 10^{-4}$
idec	7 or 8	real (BCD)	$5.000 * 10^{-9}$	$1.000 * 10^{-4}$
idec	9 or 10	real (BCD)	$5.000 * 10^{-11}$	$1.000 * 10^{-4}$
idec	11 or 12	real (BCD)	$5.000 * 10^{-13}$	$1.000 * 10^{-4}$
idec	13 or 14	real (BCD)	$5.000 * 10^{-15}$	$1.000 * 10^{-4}$
idec	15 or 16	real (BCD)	$5.000 * 10^{-17}$	$1.000 * 10^{-4}$
idec	17 or 18	real (BCD)	$5.000 * 10^{-19}$	$1.000 * 10^{-4}$
idec	> 18	real (BCD)	0	$1.000 * 10^{-4}$

E. Real Data Items

The host language variables that are consistent with various sizes of a real data item are presented in Tables V-4a and V-4b. These tables show 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. Consult Table V-4a when the PASCAL MT+ B option is used. Consult Table V-4b when the B option is not used.

With the B option: When storing data values into a real data item, there is no potential for overflow. When retrieving data from a real data item, overflow occurs if the stored data value has an absolute value greater than $1.000 * 10^{14}$.

Without the B option: When storing data values into a real data item there is no potential for overflow. When retrieving data from a real data item, overflow occurs if the stored data value has an absolute value greater than $1.844 * 10^{19}$.

F. Repeating Data Items

When storing data into or retrieving data from a repeating data item, a PASCAL MT+ 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 a data item in the DDL specification (with an occurs clause).

Repeating Data Item (m replications)		
MDBS type	MDBS size	Form of the PASCAL MT+ Variable Array:
binary	n	packed array [1:m,1:n] of char
character	n	packed array [1:m,1:n] of char
string	n	packed array [1:m,n] of string
date	-	packed array [1:m,1:10] of char
time	-	packed array [1:m,1:9] of char
integer	n	array [1:m] of integer
unsigned	1	array [1:m] of byte
unsigned	n ≥ 2	array [1:m] of word
idec(with B)	n	array [1:m] of real
idec(without B)	n	array [1:m] of real
real(with B)	n	array [1:m] of real
real(without B)	n	array [1:m] of real

VI. CONTROL PROCEDURES

A. Running an Application Program

The following steps are used to control the selective interfacing of MDBS.DMS routines with a PASCAL MT+ 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 {$I} command with an edited copy of FNDMS.PAS (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,

   ```
   MTPLUS PRG
   ```

   where PRG is the file containing the PASCAL MT+ program source code. (If you compile with the $B option then BCDMT replaces FPMT and BCDREALS.ERL replaces FPREALS.ERL in step 3 below.)

3. **If you were able to create CPMDMS,** then selectively link your compiled program and MDBS.DMS together in the following way:

   ```
   MLINK -Z -X PRG.ERL FPREALS.ERL FPMT -LCPMDMS -LPASLIB.ERL
   ```

   where PRG.ERL contains the compiled program. This assumes that the program is on the working disk that contains FPMT.REL, BCDMT.REL and CPMDMS.REL (or CPMRTL.REL in the case of RTL). It also assumes that this disk is on the default drive. For version 5.5 and greater, insert PINI.ERL immediately after FPMT and, if overlays are to be used, insert -LFRERAM.REL immediately after PINI.ERL.

   If you were unable to create CPMDMS (because you do not have LIB), then use the following command line:

   ```
   MLINK -Z -X PRG.ERL FPREALS.ERL FPMT -LDMS -LOS -LA -LPASLIB.ERL
   ```

   where PRG.ERL contains the compiled program. This assumes that the program is on the working disk that contains DMS.REL (or RTL.REL in the case of RTL), FPMT.REL, BCDMT.REL, OS.REL, the MLINK files, and A.REL. It also assumes that this disk is on the default drive. For version 5.5 and greater, insert PINI.ERL immediately after FPMT and, if overlays are to be used, insert -LFRERAM.REL immediately after PINI.ERL.

4. Execute the linked program.

C. Alternative Control Procedures

Prior to release 1.06, MLINK did not support overlays in PASCAL MT+. See Appendix C for an interfacing method that will allow you to use overlays without an MLINK command line. If you use this alternative installation procedure, then the following steps are used to execute an application program.

This alternative procedure is valuable even if overlays are not used, because it typically results in faster link times.

If using FPMT.REL:

1. Create your application program as usual, except be sure that the following two statements appear before the invocation of any DML command:

```
external procedure dbinit;
dbinit;
```

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

```
MTPLUS PRG
```

   where PRG.PAS is a file containing PASCAL MT+ source code.

3. Now resolve unsatisfied externals with LINKMT:

```
LINKMT PRG=PRG,DMS7000,FPREALS,MTJP,PASLIB/S
```

   For version 5.5 and greater, insert PINI.ERL immediately prior to PASLIB/S.

4. Execute the compiled program

```
DMS7000.COM PRG
```

If using BCDMT.REL:

1. Create your application program as usual, except be sure that the following two statements appear before the invocation of any DML command:

```
external procedure dbinit;
dbinit;
```

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

```
MTPLUS PRG $B
```

   where PRG.PAS is a file containing PASCAL MT+ source code.

VII. DML COMMAND FORM

PASCAL MT+ 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 integer.

All DML commands used in a PASCAL MT+ program must be defined as externals. The external declarations for the DML commands appear in Appendix B. These declarations are also furnished on the FNDMS.PAS file. This file can be used in conjunction with the PASCAL MT+ $I command. This is an alternative to explicitly specifying external definitions. It is accomplished with the following declarations:

```
        .
        .
        .
var e0: integer;
        .
        .
        {$I FNDMS.PAS}
```

We recommend editing a copy of FNDMS.PAS to exclude those commands that are not used by the program. Otherwise, the resulting .COM file will be larger than necessary. Of course, the name of the edited file is used with the $I command.

Program records that are to be used by DML commands within an application program should be declared at the beginning of the program. One additional record is then declared; we shall call it **rtyp** in all examples that follow. Suppose that three program records (ablk, bblk, cblk) have been declared, then **rtyp** is declared as follows:

```
rtyp = record
       case integer of
       1:(ityp:^integer);
       2:(atyp:^ablk);
       3:(btyp:^bblk);
       4:(ctyp:^cblk);
end;
intptr = ^integer;
```

The first three lines of this declaration are required; there is one aditional line for each program record previously declared. After **rtyp** is declared, you must then declare **intptr** to be of the ^integer type.

We recommend using ALTEOS, because it simplifies command status error checking. ALTEOS has the effect of changing the 255 end-of-set message to -1. For example

```
e0:=alteos
```

C. Open and Close Command Examples

```
type opent = record dbu:string[16];
                    dbp:string[12];
                    dbo:string[4];
                    dbf:string[14];
     end;
      .
      .
      .
     rtyp = record
            case integer of
            1:(ityp:^integer);
            2:(otyp:^opent);
            .
            .
            .
     end;
     intptr = ^integer;
var  e0:integer;
     optr:^opent;
     r:rtyp;
{$I FNDMS.PAS}
     .
     .
     .
     new (optr);
     r.otyp:=optr;
     e0:=dbopn (r.ityp);
         .
         .
         .
     e0:=dbcls;
```


E. Assignment Command Examples

```
type
      .
      .
      .
     rtyp = record
            case integer of
            1:(ityp:^integer);
            2:(otyp:^opent);
            3:(utyp:^byte);
            .
            .
            .
     end;
     intptr = ^integer;
var e0:integer;
    r:rtyp;
    uptr:^word;
    .
    .
    .
{$I FNDMS.PAS}
    .
    .
    .
 new (uptr);
    .
    .
    .
 r.utyp:=uptr;
 e0:=smu('setl',r.ityp);
    .
    .
    .
 e0:=som('setl,set2')
```

Here, setl 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 line is used:

```
CBRU
```

Appendix A

Syntax of DML Commands

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(r.ityp)
e0:=CCU(r.ityp)
e0:=CRA('record,area')
e0:=CRS('record',r.ityp)
e0:=DBCLS
e0:=DBCLSA('area')
e0:=DBENV(r.ityp)
e0:=DBOPN(r.ityp)
e0:=DBOPNA('area',r.ityp)
e0:=DBSAVE
e0:=DBSTAT(r.ityp)
e0:=DRC
e0:=DRM('set')
e0:=DRO('set')
e0:=FDRK('record',r.ityp)
e0:=FFM('set')
e0:=FFO('set')
e0:=FFS(['area'])
e0:=FLM('set')
e0:=FLO('set')
e0:=FMI('item,set',r.ityp)
e0:=FMSK('set',r.ityp)
e0:=FNM('set')
e0:=FNMI('item,set',r.ityp)
e0:=FNMSK('set',r.ityp)
e0:=FNO('set')
e0:=FNOI('item,set',r.ityp)
e0:=FNOSK('set',r.ityp)
e0:=FNS(['area])
e0:=FOI('item,set',r.ityp)
e0:=FOSK('set',r.ityp)
e0:=FPM('set')
e0:=FPO('set')
e0:=FRK('record',r.ityp)
e0:=GETC(r.ityp)
e0:=GETM('set',r.ityp)
e0:=GETO('set',r.ityp)
e0:=GFC('item',r.ityp)
e0:=GFM('item,set',r.ityp)
e0:=GFO('item,set',r.ityp)
e0:=GMC('set',r.ityp)
e0:=GOC('set',r.ityp)
e0:=GTC(r.ityp)
e0:=GTM('set',r.ityp)
e0:=GTO('set',r.ityp)
e0:=IMS('set')
e0:=IOS('set')
e0:=LGCPLX
e0:=LGENDX
e0:=LGFILE(r.ityp)
e0:=LGFLSH
e0:=LGMSG(r.ityp)
e0:=MAU(r.ityp)
e0:=MCC(r.ityp)
e0:=MCF
e0:=MCP
e0:=MRTF(['record'])
e0:=MRTP('record')
e0:=MSF(['set'])
e0:=MSP('set')
e0:=NCI
e0:=PFC('item',r.ityp)
e0:=PFM('item,set',r.ityp)
e0:=PFO('item,set',r.ityp)
e0:=PIFD(r.ityp)
e0:=PUTC(r.ityp)
e0:=PUTM('set',r.ityp)
e0:=PUTO('set',r.ityp)
e0:=RMS('set')
e0:=ROS('set')
e0:=RSM('set')
e0:=RSO('set')
e0:=SCM('set')
e0:=SCN
e0:=SCO('set')
e0:=SCU(r.ityp)
e0:=SETPBF(r.ityp,size)
e0:=SMC('set')
e0:=SME('set')
e0:=SMM('setl,set2')
e0:=SMN('set')
e0:=SMO('setl,set2')
e0:=SMU('set',r.ityp)
e0:=SOC('set')
e0:=SOE('set')
e0:=SOM('setl,set2')
e0:=SON('set')
e0:=SOO('setl,set2')
e0:=SOU('set',r.ityp)
e0:=SUC(r.ityp)
e0:=SUM('set',r.ityp)
e0:=SUN(r.ityp)
e0:=SUO('set',r.ityp)
e0:=SUU(r.ityp)
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(r.ityp)
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 for DML Commands

Appendix B

The following list of external declarations is included for your convenience. It is recommended that you edit a copy of this list and include ($I) it in your program.

```
external function alteos: integer;
external function amm(a:string): integer;
external function amo(a:string): integer;
external function aom(a:string): integer;
external function aoo(a:string): integer;
external function aui(a:intptr): integer;
external function ccu(a:intptr): integer;
external function cra(a:string; b:intptr): integer;
external function crs(a:string; b:intptr): integer;
external function dbcls: integer;
external function dbclsa(a:string): integer;
external function dbenv(a:intptr): integer;
external function dbopn(a:intptr): integer;
external function dbopna(a:string; b:intptr): integer;
external function dbsave: integer;
external function dbstat(a:intptr): integer;
external function drc: integer;
external function drm(a:string): integer;
external function dro(a:string): integer;
external function fdrk(a:string; b:intptr): integer;
external function ffm(a:string): integer;
external function ffo(a:string): integer;
external function ffs(a:string): integer;
external function flm(a:string): integer;
external function flo(a:string): integer;
external function fmi(a:string; b:intptr): integer;
external function fmsk(a:string; b:intptr): integer;
external function fnm(a:string): integer;
external function fnmi(a:string; b:intptr): integer;
external function fnmsk(a:string; b:intptr): integer;
external function fno(a:string): integer;
external function fnoi(a:string; b:intptr): integer;
external function fnosk(a:string; b:intptr): integer;
external function fns(a:string): integer;
external function foi(a:string; b:intptr): integer;
external function fosk(a:string; b:intptr): integer;
external function fpm(a:string): integer;
external function fpo(a:string): integer;
external function frk(a:string; b:intptr): integer;
external function getc(a:intptr): integer;
external function getm(a:string; b:intptr): integer;
external function geto(a:string; b:intptr): integer;
external function gfc(a:string; b:intptr): integer;
external function gfm(a:string; b:intptr): integer;
external function gfo(a:string; b:intptr): integer;
external function gmc(a:string; b:intptr): integer;
external function goc(a:string; b:intptr): integer;
external function gtc(a:intptr): integer;
```


Appendix C

Alternative MDBS.DMS Interfacing Method

Appendix C

A solution to the overlay problem is to create a single DMS runtime module for all programs in an application system. This DMS runtime module is loaded before application programs are run. It resides in its own portion of memory, protected from application programs. The DMS runtime module is created by the interactive MKDMS.

A. Installation

Copy MKDMS, MKTAB, the MLINK files, MTJP.ERL, MTJP2.REL, DBRUN.REL, DMS.REL, OS.REL, A.REL, Z.REL, and FPMT.REL or BCDMT.REL to the same working disk. If you intend to use any of the special link files (e.g., NOCALC), they should also be copied to the working disk; you should not use any of the special link files with the sample program.

Now execute MKDMS. This interactive installation program generates a command file which will create a DMS runtime module that supports desired DML commands. Appendix D shows fifteen predefined DML command groupings. One of these groupings can be selected or a customized group of DML commands can be specified during interaction with MKDMS.

When MKDMS prompts for a response, the permissible responses are indicated in parentheses and the default response appears in square brackets. Pressing the ENTER key (alone) yields the default response. Below is a sample session using MKDMS. In this example, the command grouping R2W1 is selected and the default command file name of DMS7000 is assigned to drive B. Operator responses are shown in bold type.

```
MKDMS v1.00
(C) COPYRIGHT 1982, Micro Data Base Systems, Inc.

This program creates a command file that will create a DMS
'runtime' module supporting selected DML commands.

First, you must determine where in memory the DMS runtime
module will reside.  This is called the ORG address (org is
derived from the word origin by the usual means of dropping
various letters from a word until it is barely
decipherable).  You should org the DMS runtime module as
high as possible in memory.  If you are not sure where to
org it, pick a nice high address (e.g., A000).  If the
address is too high, the loader that MKDMS prefixes to the
runtime module will inform you as to the highest org address
possible for that runtime module.  Currently the average
runtime module size is about 22k or 5800h.

Enter desired org address (in hex): 7000                    *
```

---

* Note: This org address is hardware/operating system dependent; 7000 is workable in most environments, but if this address fails another address should be chosen. Select a very high address. If it is too high, you will be informed as to the maximum address that can be used in your environment. In general it is advisable to use this maximum address.

Finally, there is the issue of where to find the 'utility' files needed by MKDMS. The files in question are:

MLINK.COM, MKTAB.COM, DBRUN.REL, DMS.REL, OS.REL, A.REL, Z.REL, FPMT.REL, AND MTJP2.REL

These files will be used to construct your DMS runtime module, when you execute the command file.

Enter utilities drive, if any:

One moment whilst MKDMS generates the command file ...

There now exists a file (b:DMS7000.SUB) which contains the commands that will create your DMS runtime module. To execute this command file, enter the following command:
SUBMIT b:DMS7000.SUB

Now, to create the DMS runtime routine, you will execute the command file created by MKDMS. This command file invokes the MLINK and MKTAB programs:

SUBMIT B:DMS7000.SUB

If using floating point (FPMT.REL), you will see the following:

```
MDBS MLINK V1.03c (Z80)
(C) COPYRIGHT 1982, Micro Data Base Systems, Inc.
Lafayette, IN  47902

DBRUN:
     DBRUN
1 modules

MTJP2:
     MTJP
1 modules

FPMT:
     DDMDBS UNDEF  ALTEOS VARCS  LNPSMT GETSP  GETFWA EIINT
     IEINT  EIREAL IEREAL EISTR  IESTR  EICHAR IECHAR
15 modules

DMS:
     CRS    FDRK   F2MDBS FRK    IMS    RCMDBS DBSTAT FDMDBS
     IFMDBS PJMDBS SMC    DBOPN  DSMDBS FEMDBS YLMDBS GETC
     GFC    GUMDBS IJMDBS F8MDBS YKMDBS YMMDBS PNMDBS EBMDBS
     DCMDBS POMDBS IKMDBS DEMDBS BBMDBS C8MDBS YIMDBS YJMDBS
     DBCLS  DBSAVE ECMDBS GDMDBS IIMDBS PGMDBS GOMDBS RAMDBS
     SCM    SCR    SRC    UAMDBS ACMDBS PUMDBS DQMDBS MNMDBS
     PAMDBS UHMDBS BDMDBS NBMDBS NCMDBS SMMDBS C9MDBS YNMDBS
     SJMDBS SKMDBS SLMDBS ZBMDBS CDMDBS GNMDBS GSMDBS SOMDBS
     ZAMDBS NDMDBS NFMDBS NGMDBS
68 modules
```

If using BCD (i.e., BCDMT.REL), you will see the following:

```
SUBMIT B:DMS7000.SUB

MDBS MLINK V1.03c (Z80)
(C) COPYRIGHT 1982, Micro Data Base Systems, Inc.
Lafayette, IN  47902

DBRUN:
     DBRUN
1 modules

MTJP2:
     MTJP
1 modules

BCDMT:
     DDMDBS UNDEF  ALTEOS VARCS  LNPSMT GETSP  GETFWA EIINT
     IEINT  EIREAL IEREAL EISTR  IESTR  EICHAR IECHAR
15 modules

DMS:
     CRS    FDRK   F2MDBS FRK    IMS    RCMDBS DBSTAT FDMDBS
     IFMDBS PJMDBS SMC    DBOPN  DSMDBS FEMDBS YLMDBS GETC
     GFC    GUMDBS IJMDBS F8MDBS YKMDBS YMMDBS PNMDBS EBMDBS
     DCMDBS POMDBS IKMDBS DEMDBS BBMDBS C8MDBS YIMDBS YJMDBS
     DBCLS  DBSAVE ECMDBS GDMDBS IIMDBS PGMDBS GOMDBS NAMDBS
     RAMDBS SCM    SCR    SRC    UAMDBS ACMDBS PUMDBS DQMDBS
     MNMDBS PAMDBS UHMDBS BDMDBS NBMDBS NCMDBS SMMDBS C9MDBS
     YNMDBS SJMDBS SKMDBS SLMDBS ZBMDBS CDMDBS GNMDBS GSMDBS
     SOMDBS ZAMDBS NDMDBS NFMDBS NGMDBS
69 modules

OS:
     FCB    DBFCBL DBPRUL MFNLEN DFOP   FNAME  SCRFCB DFSK
     DFWR   DFCL   CASCVT SYSCPM BDOS
13 modules

A:
     DIVZ8  DIVZ16 MULZ16 MULZ8
4 modules

Z:
     ZZZLWA
1 modules
Code = 6C00 Common = C010 Data = C2BF
Next = C409

DBRUN:
     DBRUN

MTJP2:
     MTJP

BCDMT:
     DDMDBS UNDEF  ALTEOS VARCS  LNPSMT GETSP  GETFWA EIINT
     IEINT  EIREAL IEREAL EISTR  IESTR  EICHAR IECHAR
```


Appendix D

DML Retrieval/Write Command Groups

Appendix D

R1W0

alteos	extend	fmsk	frk	smm	sor	varcs
dbcls	fdrk	fnm	scm	smo	src	
dbopn	ffm	fno	sco	smr	srm	
dbsave	ffo	fosk	scr	soc	sro	
dbstat	flm	fpm	setpbf	som	undef	
define	flo	fpo	smc	soo	varcmd	

R2W0

alteos	fdrk	fno	geto	sco	soc	undef
dbcls	ffm	fosk	getr	scr	som	varcmd
dbopn	ffo	fpm	gfc	setpbf	soo	varcs
dbsave	flm	fpo	gfm	smc	sor	
dbstat	flo	frk	gfo	smm	src	
define	fmsk	getc	gfr	smo	srm	
extend	fnm	getm	scm	smr	sro	

R3W0

alteos	define	fnmi	geto	scn	soe	suo
aui	extend	fnmsk	getr	sco	som	suu
cct	fdrk	fno	gfc	scr	son	tcn
ccu	ffm	fnoi	gfm	scu	soo	tct
cmt	ffo	fnosk	gfo	setpbf	sor	tmn
cot	ffs	fns	gfr	smc	sou	tmt
dbcls	findm	foi	gmc	sme	src	toggle
dbclsa	findo	fosk	goc	smm	srm	ton
dbenv	flm	fpm	gtc	smn	srn	tot
dbopn	flo	fpo	gtm	smo	sro	tun
dbopna	fmi	frk	gto	smr	suc	undef
dbsave	fmsk	getc	nci	smu	sum	varcmd
dbstat	fnm	getm	scm	soc	sun	varcs

R1W2

alteos	extend	fosk	lgflsh	puto	sfr	src
cr	fdrk	fpm	lgmsg	putr	smc	srm
cra	ffm	fpo	pfc	scm	smm	sro
crs	ffo	frk	pfm	sco	smo	trabt
dbcls	flm	ims	pfo	scr	smr	trbgn
dbopn	flo	ios	pfr	setpbf	soc	trcom
dbsave	fmsk	lgcplx	pifd	sfc	som	undef
dbstat	fnm	lgendx	putc	sfm	soo	varcmd
define	fno	lgfile	putm	sfo	sor	varcs

R2W2

alteos	ffm	getc	lgendx	puto	smm	trbgn
cr	ffo	getm	lgfile	putr	smo	trcom
cra	flm	geto	lgflsh	scm	smr	undef
crs	flo	getr	lgmsg	sco	soc	varcmd
dbcls	fmsk	gfc	pfc	scr	som	varcs
dbopn	fnm	gfm	pfm	setpbf	soo	
dbsave	fno	gfo	pfo	sfc	sor	
dbstat	fosk	gfr	pfr	sfm	src	
define	fpm	ims	pifd	sfo	srm	
extend	fpo	ios	putc	sfr	sro	
fdrk	frk	lgcplx	putm	smc	trabt	

R3W2

alteos	extend	fns	gtm	puto	smr	suu
aui	fdrk	foi	gto	putr	smu	tcn
cct	ffm	fosk	ims	scm	soc	tct
ccu	ffo	fpm	ios	scn	soe	tmn
cmt	ffs	fpo	lgcplx	sco	som	tmt
cot	findm	frk	lgendx	scr	son	toggle
cr	findo	getc	lgfile	scu	soo	ton
cra	flm	getm	lgflsh	setpbf	sor	tot
crs	flo	geto	lgmsg	sfc	sou	trabt
dbcls	fmi	getr	nci	sfm	src	trbgn
dbclsa	fmsk	gfc	pfc	sfo	srm	trcom
dbenv	fnm	gfm	pfm	sfr	srn	tun
dbopn	fnmi	gfo	pfo	smc	sro	undef
dbopna	fnmsk	gfr	pfr	sme	suc	varcmd
dbsave	fno	gmc	pifd	smm	sum	varcs
dbstat	fnoi	goc	putc	smn	sun	
define	fnosk	gtc	putm	smo	suo	

R1W4

alteos	dbstat	fmsk	lgendx	putr	sfr	trabt
amm	define	fnm	lgfile	rms	smc	trbgn
amo	drc	fno	lgflsh	ros	smm	trcom
ams	drm	fosk	lgmsg	rsm	smo	undef
aom	dro	fpm	pfc	rso	smr	varcmd
aoo	drr	fpo	pfm	scm	soc	varcs
cr	extend	frk	pfo	sco	som	xmm
cra	fdrk	getr	pfr	scr	soo	xmo
crs	ffm	gffr	pifd	setpbf	sor	xom
dbcls	ffo	ims	putc	sfc	src	xoo
dbopn	flm	ios	putm	sfm	srm	
dbsave	flo	lgcplx	puto	sfo	sro	

R2W4

alteos	drc	fpm	lgendx	ros	soc	xmo
amm	drm	fpo	lgfile	rsm	som	xom
amo	dro	frk	lgflsh	rso	soo	xoo
ams	drr	getc	lgmsg	scm	sor	
aom	extend	getm	pfc	sco	src	
aoo	fdrk	geto	pfm	scr	srm	
cr	ffm	getr	pfo	setpbf	sro	
cra	ffo	gfc	pfr	sfc	trabt	
crs	flm	gfm	pifd	sfm	trbgn	
dbcls	flo	gfo	putc	sfo	trcom	
dbopn	fmsk	gfr	putm	sfr	undef	
dbsave	fnm	ims	puto	smc	varcmd	
dbstat	fno	ios	putr	smm	varcs	
define	fosk	lgcplx	rms	smr	xmm	

R3W4

alteos	dbsave	fnmi	gmc	putm	smm	suo
amm	dbstat	fnmsk	goc	puto	smn	suu
amo	define	fno	gtc	putr	smo	tcn
ams	drc	fnoi	gtm	rms	smr	tct
aom	drm	fnosk	gto	ros	smu	tmn
aoo	dro	fns	ims	rsm	soc	tmt
aui	drr	foi	ios	rso	soe	toggle
cct	extend	fosk	lgcplx	scm	som	ton
ccu	fdrk	fpm	lgendx	scn	son	tot
cmt	ffm	fpo	lgfile	sco	soo	trabt
cot	ffo	frk	lgflsh	scr	sor	trbgn
cr	ffs	getc	lgmsg	scu	sou	trcom
cra	findm	getm	nci	setpbf	src	tun
crs	findo	geto	pfc	sfc	srm	undef
dbcls	flm	getr	pfm	sfm	srn	varcmd
dbclsa	flo	gfc	pfo	sfo	sro	varcs
dbenv	fmi	gfm	pfr	sfr	suc	xmm
dbopn	fmsk	gfo	pifd	smc	sum	xmo
dbopna	fnm	gfr	putc	sme	sun	xom
						xoo

Appendix E

DML Usage in PASCAL

Appendix E

Since PASCAL is very strict about type checking, those DML commands which have several different record structures as one of their parameters during the course of a program must use a record structure with a variant part. The variant part defines the record structure as consisting of different types and numbers of components depending on the context. In the interest of compact source code, an explicit tag field is not used. The record structure needed by DML commands, rtyp, is defined by

```
rtyp = record
       case integer of
       1:(ityp:^integer);
       2:(atyp:^ablk);
         .
         .
         .
       n:(ntyp:^nblk);
end;
```

where ablk, ... nblk are the record structures needed by DML to describe the data in the database.

An expression is not allowed in external function declarations. Because PASCAL considers ^integer to be an expression, it is necessary to define

```
intptr = ^integer;
```

in every program prior to the external function declarations.

It is also necessary to type the record itself and the variables in the variant part

```
var
  aptr:^ablk
   .     .
   .     .
   .     .
  nptr:^nblk
  r:rtyp
   .     .
   .     .
   .     .
```

before the body of the program.

To use this record in a DML command, set the value of the record rtyp equal to the record structure wanted, with statement

```
r.ntyp=nptr;
```

and invoke the DML command with the statement

```
e0:=CMD(string,r.ityp);
```

The PASCAL compiler only checks the type of the parameters in a function call, not the actual value of the record. Since r.ityp is of type integer, all the type checking rules are satisfied (even though the value of rtyp has been set equal to the record needed).

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.