MDBS III

SYSTEM SPECIFIC INSTALLATION MANUAL

for

CP/M WITH COMPILED BASIC (BASCOM)

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

USA

December 1981
(Revised November 1982: MDBS 3.05)

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 decypher. 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.

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 for Chaining

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:BASCOM environment, these files have the following names:

DMS.REL	MDBS.DMS library (standard form)
DBRUN.REL	runtime loader component
BASCOM80.REL	BASCOM language interface
L1BASCOM.REL	BASCOM language interface - indirect form extension
A.REL	MDBS.DMS component
OSCPM.REL	CP/M interface
Z.REL	MDBS.DMS component
DMSTAB.MAC	MDBS.DMS component
RTL.REL	MDBS.DMS library (RTL form)
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.BAS	direct call sample program
SAMPLEI.BAS	indirect call sample program
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
QRS9.OVL	QRS support overlay 9
QRS10.OVL	QRS support overlay 10
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
IDML6.OVL	IDML support overlay 6
IDML7.OVL	IDML support overlay 7
IDML8.OVL	IDML support overlay 8
DMU.COM	MDBS.DMU object code
RCV.COM	object code for the MDBS-RTL recovery program:RCV
CNV.COM	object code for the DDL conversion program:CNV
R1W0.REL	indirect DML reference table: level 1 reads, no writes
R1W1.REL	indirect DML reference table: level 1 reads, level 1 writes
R1W2.REL	indirect DML reference table: level 1 reads, level 2 writes
R1W3.REL	indirect DML reference table: level 1 reads, level 3 writes

R1W4.REL	indirect DML reference table: level 1 reads, level 4 writes
R2W0.REL	indirect DML reference table: level 2 reads, no writes
R2W1.REL	indirect DML reference table: level 2 reads, level 1 writes
R2W2.REL	indirect DML reference table: level 2 reads, level 2 writes
R2W3.REL	indirect DML reference table: level 2 reads, level 3 writes
R2W4.REL	indirect DML reference table: level 2 reads, level 4 writes
R3W0.REL	indirect DML reference table: level 3 reads, no writes
R3W1.REL	indirect DML reference table: level 3 reads, level 1 writes
R3W2.REL	indirect DML reference table: level 3 reads, level 2 writes
R3W3.REL	indirect DML reference table: level 3 reads, level 3 writes
R3W4.REL	indirect DML reference table: level 3 reads, level 4 writes
MLINK.COM	MDBS linker utility
MLINK1.OVL	MLINK overlay for pass 1
MLINK2.OVL	MLINK overlay for pass 2
MRENAME.COM	DML renaming utility
MKDMS	creates selective DMS runtime module
MKTAB	creates entry point table for DMS runtime module
FASTIO.REL	link file for overriding CP/M disk access mechanism
NOCALC.REL	link file for disabling CALC
NOFLOAT.REL	link file for disabling real and idec data manipulation
NOREAL.REL	link file for disabling real data manipulation
NOIDEC.REL	link file for disabling idec data manipulation
NODATE.REL	link file for disabling date data manipulation
NOTIME.REL	link file for disabling time data manipulation
COMDEF.REL	common block definitions for chaining
SAMPLEC1.BAS	sample program using chaining
SAMPLEC2.BAS	sample chained module
SAMPLEC3.BAS	sample chained module
SAMPLEC4.BAS	sample chained module
SAMPLEC5.BAS	sample chained module

If a license is not purchased for the RTL form of MDBS and for all MDBS add-on packages, then some of these files will not be furnished.All of the link files for disabling various MDBS features are not always available in all environments, under all releases of MDBS.

B. Fully Qualified File Names in CP/M

A fully qualified CP/M file name consists of a valid CP/M file name, optionally prefaced by a drive specification. A CP/M file name consists of from one to eight alphanumeric characters, with an optional extension of up to three alphanumeric characters. If a drive specification is used, it must be A, B, ..., or P and it must be followed by a colon. If **TRIAL.DDL** is a CP/M file name for a file on drive E, then the fully qualified file name is:

```
E:TRIAL.DDL
```

If a drive specification is omitted from a fully qualified file name, then the default drive directory is assumed. Note that any file names specified in a DDL description should be enclosed in double quotes (" "). File name (and extension) defaults are described in Chapter IV.

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 schema 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. In certain MP/M environments, it is possible to override the usual MP/M directory-access mechanism to achieve substantial increases in MDBS access speeds. Where possible, Micro Data Base Systems, Inc. will do the work necessary to override this mechanism. Contact Micro Data Base Systems, Inc. for information about this service and the fees charged for performing this service.

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.**

F. The Renaming Utility

The MRENAME.COM file contains a utility that can be used to change the names of DML commands. For instance, you can use this utility to change the name PUTC to DBPUTC. Before invoking this utility, create a file (call it DML.REN) containing the changes to be made. Each line of this file begins with the name of an existing DML command, followed by a blank space, followed by the new name of the DML command to be used in your application programs. The renaming is accomplished by entering the following operating system command line:

```
MRENAME  DMS.REL  DML.REN
```

This assumes that DMS.REL and DML.REN are on the default drive. Note that MRENAME may take a considerable period of time to execute. The output is a file called DMS.PHN. Copy DMS.PHN to a different working disk, givng it the name DMS.REL. This new DMS.REL is used in place of the DMS.REL supplied on the distribution disk.

II. INSTALLATION and TESTING PROCEDURES

A. Installation

1. MDBS.DDL

MDBS.DDL is installed by simply copying 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

MDBS.DMS can be used in either the direct or indirect forms. **If you intend to use the indirect form**, then you need to select one of the fifteen files R1W0, R1W1, ..., R3W4. Appendix B shows the DML commands included on each of these files. You should select a file that includes all DML commands which are to be used in your application program. Alternatively, you can specify a file of your own. This file is created by editing DMSTAB.MAC to eliminate any DML commands that are not needed by your application program. This edited file must then be assembled by an assembler that produces REL output. The resulting REL file can then be used in place of R1W0, R1W1, ..., or R3W4 in the installation procedure. The file that you choose or create for indirect DML usage will be referred to as RnWm.REL in the following discussion.

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

```
LIB CPMDMS = DMS,OSCPM,A,Z/E
```

This assumes that the DMS.REL, OSCPM.REL, A.REL, and Z.REL files are on the default drive. If you intend to use MDBS-RTL, the same approach is employed, except use RTL instead of DMS and CPMRTL instead of CPMDMS.

Now copy BASCOM80.REL, L1BASCOM.REL, RnWm.REL, and CPMDMS.REL (or CPMRTL.REL) 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. MLINK is invoked by:

MLINK **pgm** BASCOM80 -LCPMDMS -LBASLIB

where **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 program file used with MLINK, except it has a .COM extension. This output file will reside on the same drive as the first program file. An alternative output file name (and/or drive) 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). As indicated in the following documentation, there are cases where a -K0 argument must immediately follow MLINK.

C. Testing

After the installation of MDBS.DDL and MDBS.DMS, the sample DDL specification and sample application program can be used to test the success of the installation procedure. Two sample application programs are provided. One uses direct DML invocation (SAMPLE.BAS). The other uses the indirect form of DML invocation (SAMPLEI.BAS). **We strongly recommend the direct form.** The indirect interface is included only as a convenience for those who have developed application programs with MBASIC and want to use those same programs in the BASCOM environment with minimal alterations. We do not generally recommend the indirect form since it uses more memory, results in longer programs, and is not guaranteed to be supported in the future.

As the following examples illustrate, the method for utilizing MDBS.DMS differs depending on whether direct or indirect invocation is employed and depending on the version of BASCOM that you have.

DIRECT INVOCATION

1. To initialize the sample data base, enter:

   DDL SAMPLE.DDL

   This assumes that the DDL working disk is on the default drive.

2. To compile the sample application program, enter:

   BASCOM =B:SAMPLE/E/C

   This assumes that SAMPLE.BAS is on drive B.

3. **For BASCOM versions below 5.3** use MLINK as follows:

```
MLINK B:SAMPLE B:BASCOM80 -LB:CPMDMS -LB:BASLIB
```

For BASCOM versions 5.3 and greater use MLINK as follows

```
MLINK B:SAMPLE B:BASCOM80 -LB:CPMDMS -LB:OBSLIB
```

This assumes that you use the /O switch during compilation.

4. To execute the linked sample application program, enter:

```
B:SAMPLE
```

INDIRECT INVOCATION

1. To utilize the sample data base, enter:

```
DDL SAMPLE.DDL
```

This assumes that the DDL working disk is on the default drive.

2. To compile the sample application program, enter:

```
BASCOM =B:SAMPLEI/E
```

This assumes that SAMPLEI.BAS is on drive B.

3. The sample program needs a retrieval level of at least 2 and a write level of at least 1, so the file R2W1 is used here. These examples assume that all files to be linked are on drive B.

For BASCOM versions below 5.3 use MLINK as follows:

```
MLINK B:SAMPLEI B:L1BASCOM B:R2W1 B:BASCOM80 -LB:CPMDMS -LB:BASLIB
```

For BASCOM versions 5.3 and greater use MLINK in the same way except substitute -LB:OBSLIB for -LB:BASLIB. This assumes that you used the /O switch during compilation.

4. To execute the linked sample application program, enter:

```
B:SAMPLEI
```

D. Alternative MDBS.DMS Installation Method

Limitations of the L80 linker do not allow it to support BASCOM chaining schemes in conjunction with MDBS. There are two alternative installation methods which do allow chaining. One involves creating a DMS runtime module. This is described in Appendix C and is recommended in cases where there is sufficient memory, because it saves linking and chaining time. If there is insufficient memory for this method, the approach described in Appendix D should be used.

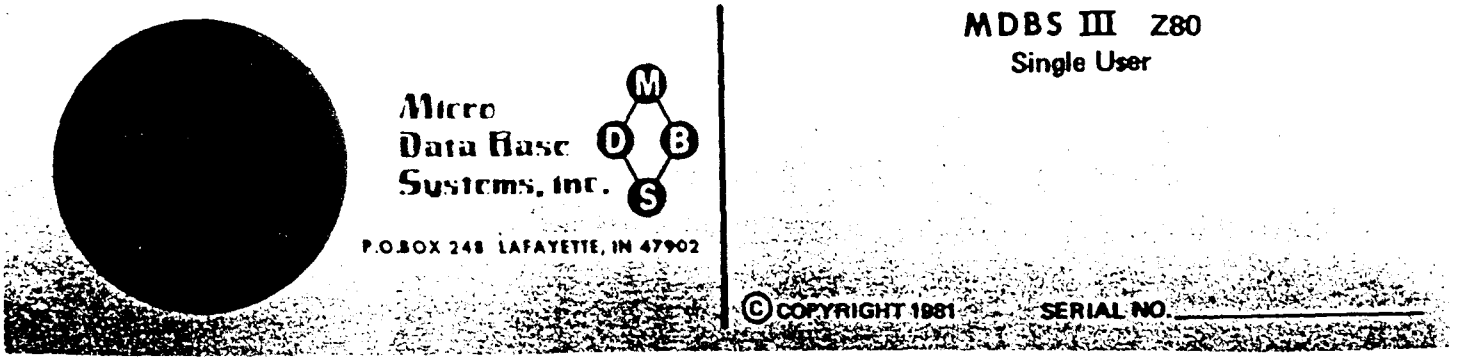
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



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.

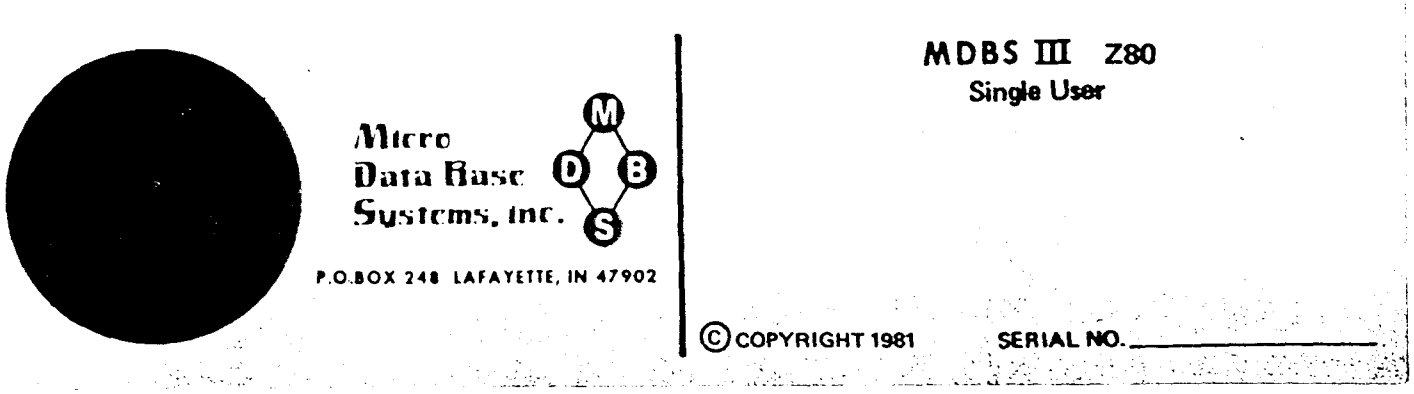
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



IV. OPERATING SYSTEM DEPENDENT DDL DEFAULTS

A. File name defaults for areas

If a file name is not specified for the main area in the identification section of a DDL specification, then the main area is assigned to a file having the same name as the data base. This file will have a .DB extension. If a file name is not specified for an area defined in the area section of a DDL specification, then that area is assigned to a file having the same name as the area. This file will have a .DBA extension. The default file names will be in upper case.

B. File extension defaults

If a file name is specified without an extention a default extension is used. The default extensions are:

for a main area file	.DB
for an extra area file	.DBA
for a file written by MDBS.DDL	.DDL
for RCV scratch files	.$$$
for a page image file	.PIF
for a transaction log file	.LOG
for QRS WRITE or SPEW files	.TXT

In order to suppress the default extension for a file, the fully qualified file name should end with a decimal point.

C. Pages per area default

If the number of pages for an area is not stated in a DDL specification, then the default value of 50 pages is used. This default is for both the main area and any extra areas.

D. Page size default

If the page size for an area is not stated in a DDL specification, then the default value of 512 bytes per page is used. This default is for both the main area and any extra areas.

E. Page size restrictions

The minimum page size for any area is 256 bytes.

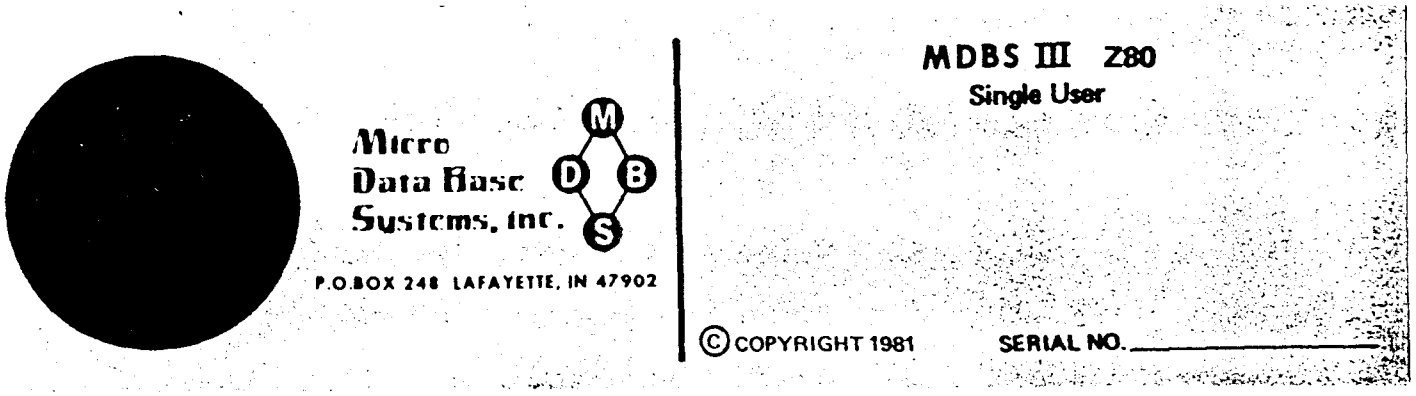
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



V. DATA ITEM - HOST LANGUAGE VARIABLE CORRESPONDENCE

This chapter shows the type, size, and value correspondences that exist between MDBS data items and BASCOM 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 BASCOM 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		BASCOM Variable	
MDBS type	MDBS size	BASCOM type	BASCOM size
binary	n	string(e.g.,A$)	n
character	n	string	n
string	n	string	
date	-	string	10
time	-	string	9

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 BASCOM 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 BASCOM variables during data retrieval (e.g., GFM) are shown.

As an example, when storing a data value from a BASCOM integer variable into a one byte integer data item, the value must be in the range -128 to 127. Any other value for the BASCOM 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 BASCOM integer variable, an appropriate value in the range -32768 to 32767 is deposited in the BASCOM variable. If the stored value is outside of this range, then the contents of the BASCOM 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 BASCOM integer variable having the value 32700.

Table V-1. Integer Correspondences

MDBS Data Item		BASCOM Variable	Storing Data		Retrieving Data	
MDBS type	MDBS size	BASCOM type	Source: BASCOM variable range	Target: data item range	Source: data item range	Target: BASCOM variable range
integer	1	integer (e.g., A%)	-32768 to -129 -128 to 127 128 to 32767	undefined -128 to 127 undefined	-128 to 127	-128 to 127
integer	2	integer (e.g., A%)	-32768 to 32767	-32768 to 32767	-32768 to 32767	-32768 to 32767
integer	n>2	integer (e.g., A%)	-32768 to 32767	-32768 to 32767	$\leq$ -32769 -32768 to 32767 $\geq$ 32768	undefined -32768 to 32767 undefined

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 BASCOM 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 BASCOM variables during data retrieval (e.g., GETM) are shown.

As an example, when storing a data value from BASCOM integer variable into a one byte unsigned data item, the value must be in the range Ø to 255. If the value is in the range 256 to 65535 or the range -32768 to -1, it will not be permitted and the DML command that attempts to store such a value will return a command status of 33. When retrieving the value 65534 from a 3 byte unsigned data item into a BASCOM integer variable, the value -2 is deposited into the BASCOM variable. As a third example, suppose we want to store the value 32770 into a two byte unsigned data item. This is accomplished with a put command that uses a BASCOM integer variable having the value -32766.

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 eight digits (i.e., $n \leq 8$), overflow occurs if the stored data value has an absolute value greater than 17014117838986683081876969772907l284224. When retrieving data from an idec data item whose size exceeds eight digits (i.e., $n > 8$), overflow occurs if the stored data value has an absolute value greater than 17014117838986683081876969772907l284224.

There is little incentive to use internal decimal data items when the primary host languages are BASCOM and FORTRAN. Internal decimal interfaces are present to permit communication (through BASCOM) to data bases that are also used with host languages such as COBOL.

Table V-2. Unsigned Correspondences

MDBS Data Item		BASCOM Variable	Storing Data		Retrieving Data	
MDBS type	MDBS size	BASCOM type	Source: BASCOM variable range	Target: data item range	Source: data item range	Target: BASCOM variable range
unsigned	1	integer (e.g., A%)	-32768 to -1 0 to 255 256 to 32767	undefined 0 to 255 undefined	0 to 255	0 to 255
unsigned	2	integer (e.g., A%)	-32768 to -1 0 to 32767	32768 to 65535 0 to 32767	0 to 32767 32768 to 65535	0 to 32767 -32768 to -1
unsigned	n>2	integer (e.g., A%)	-32768 to -1 0 to 32767	32768 to 65535 0 to 32767	0 to 32767 32768 to 65535 ≥ 65536	0 to 32767 -32768 to -1 undefined

Table V-3. Internal Decimal Correspondences

MDBS Data Item		BASCOM Variable	Storing Data	Retrieving Data
MDBS type	MDBS size	BASCOM type	Largest Relative Error	Largest Relative Error
idec	1 or 2	single precision (e.g., A!)	$5.000 * 10^{-3}$	$2.980 * 10^{-8}$
idec	3 or 4	single precision	$5.000 * 10^{-5}$	$2.980 * 10^{-8}$
idec	5 or 6	single precision	$5.000 * 10^{-7}$	$2.980 * 10^{-8}$
idec	7 or 8	single precision	$5.000 * 10^{-9}$	$2.980 * 10^{-8}$
idec	9 or 10	double precision (e.g., A#)	$5.000 * 10^{-11}$	$6.94 * 10^{-18}$
idec	11 or 12	double precision	$5.000 * 10^{-13}$	$6.94 * 10^{-18}$
idec	n > 12	double precision	$\frac{10^{-2[\frac{n+1}{2}]}}{2}$	$6.94 * 10^{-18}$

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 $1.85 * 10^{38}$ into a three byte real data item will result in an undefined value for that data item value.

Table V-4. Real Correspondences

MDBS Data Item		BASCOM Variable	Storing Data		Retrieving Data	
MDBS type	MDBS size	BASCOM type	Absolute Value Beyond Which Overflow Occurs	Largest Relative Error	Absolute Value Beyond Which Overflow Occurs	Largest Relative Error
real	2	single precision (e.g. A!)	$1.698 * 10^{38}$	$1.953 * 10^{-3}$	no overflow	0
real	3	single precision	$1.701 * 10^{38}$	$7.629 * 10^{-6}$	no overflow	0
real	4	single precision	no overflow	0	no overflow	0
real	5	double precision (e.g. A#)	$1.701 * 10^{38}$	$1.164 * 10^{-10}$	no overflow	0
real	6	double precision	$1.701 * 10^{38}$	$4.549 * 10^{-13}$	no overflow	0
real	7	double precision	$1.701 * 10^{38}$	$1.770 * 10^{-15}$	no overflow	0
real	8	double precision	no overflow	0	no overflow	0
real	>8	double precision	no overflow	0	$1.701 * 10^{38}$	$6.94 * 10^{-18}$

F. Repeating Data Items

When storing data into or retrieving data from a repeating data item, a BASCOM 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 data item in the DDL specification (with an occurs clause). Here, A is the host language array being used for storage or retrieval. **This assumes that all array subscripting begins at Ø (not 1).**

Repeating Data Item (m replications)		Form of the BASCOM Variable Array: A
MDBS type	MDBS size	
binary	n	DIM A$(m-1)
character	n	DIM A$(m-1)
string	n	DIM A$(m-1)
date	-	DIM A$(m-1)
time	-	DIM A$(m-1)
integer	n	DIM A%(m-1)
unsigned	n	DIM A%(m-1)
idec	n=1,2,...,or 8	DIM A!(m-1)
idec	n ≥ 9	DIM A#(m-1)
real	n=2,3,or4	DIM A!(m-1)
real	n ≥ 5	DIM A#(m-1)

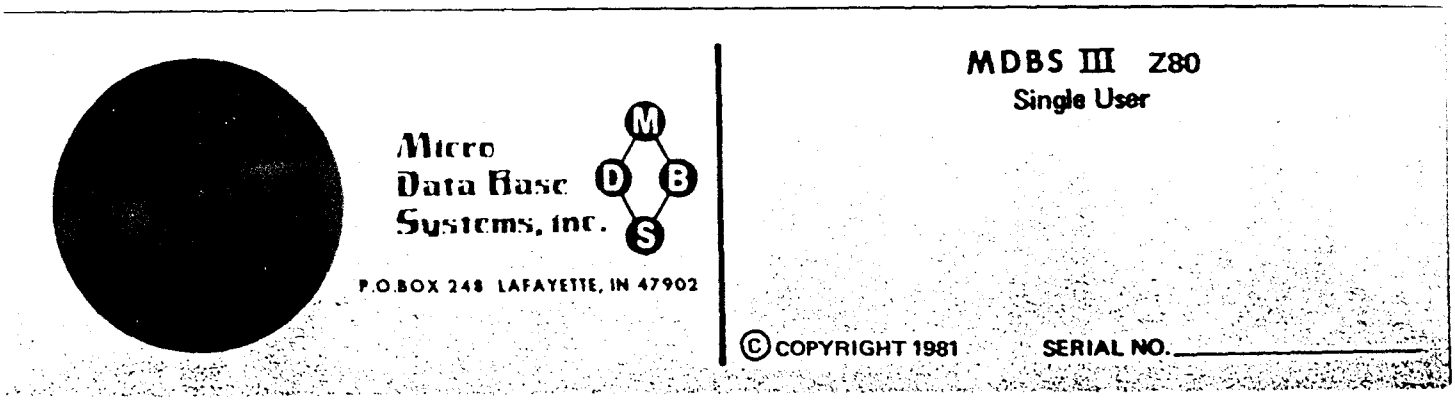
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



VI. CONTROL PROCEDURES

A. Running an Application Program

The following steps are used to control the selective interfacing of MDBS.DMS routines with a BASCOM application program. They assume that CPMDMS (or CPMRTL) has been created as described in Chapter II.

1. Create your application program using either the direct or indirect DML command form (see Chapter VII). We strongly recommend the direct form. Continuing support of the indirect form is not guaranteed.

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

   BASCOM PRG

   where PRG is the file containing the BASCOM program source code.

3. Selectively link your compiled program and MDBS.DMS together. This assumes that all files to be linked are on the working disk. It also assumes that this disk is on the default drive.

   DIRECT INVOCATION

   For BASCOM versions below 5.3 use MLINK as follows:

   MLINK PRG BASCOM80 -LCPMDMS -LBASLIB

   For BASCOM versions 5.3 and greater, compile with the /O switch and use the same MLINK line except -LOBSLIB replaces -LBASLIB.

   INDIRECT INVOCATION

   For BASCOM versions below 5.3 use MLINK as follows:

   MLINK PRG L1BASCOM RnWm BASCOM80 -LCPMDMS -LBASLIB

   For BASCOM versions 5.3 and greater, compile with the /O switch and use the same MLINK line except -LOBSLIB replaces -LBASLIB.

4. Execute the linked program:

   PRG

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 = DMS,FASTIO,OSCPM,A,Z/E
```

instead of the command line shown in Chapter II. If NOCALC is desired, then it is linked by inserting NOCALC immediately prior to BASCOM80 in the MLINK command line. If you desire to link any of the other special files, then insert the file name(s) prior to BASCOM80 and use -LBASCOM80 in place of BASCOM80 on the MLINK 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.

If FASTIO is linked in, then the data base control system will not use conventional CP/M disk access facilities. Instead, a faster disk I/O mechanism incorporating buffering of FCBs is used. Due to various environments' peculiarities, FASTIO is not guaranteed to function or improve performance in all possible CP/M and "CP/M-like" environments. Its main advantage is realized with version 2.2 CP/M.

If NOCALC is linked in, then manipulation of calc records is prohibited. This can save space (typically about 650 bytes), resulting in faster processing. NOCALC is very valuable when calc records are not used.

If NOFLOAT is linked in, the manipulation of real and idec data items is prohibited. This can save space (typically about 1150 bytes) and results in faster processing. NOFLOAT is valuable when no idec or real data item processing is needed.

If NOREAL is linked in, the manipulation of real data items is prohibited. This may save space, resulting in faster processing. NOREAL is valuable when no real data item processing is needed.

If NOIDEC is linked in, the manipulation of idec data items is prohibited. This may save space, resulting in faster processing. NOIDEC is valuable when no idec data item processing is needed.

If NODATE is linked in, the manipulation of date data items is prohibited. This can save space (typically about 450 bytes), resulting in faster processing. NODATE is valuable when no date data item processing is needed.

If NOTIME is linked in, the manipulation of time data items is prohibited. This can save space (typically about 170 bytes) and results in faster processing. NOTIME is valuable when no time data item processing is needed.

C. Alternative Control Procedures for Chaining

If the alternative installation method described in Appendix C is used, rather than the above methods, then the following control procedure is used for running an application program.

1. Create your application program using the direct DML command form.

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

   ```
   BASCOM =PRG/E/C
   ```

   where PRG is the file containing the BASCOM program source code.

3. Now resolve unsatisfied externals as follows:

   ```
   L80 PRG,DMS7000,PRG/N/E
   ```

4. Run the program by executing DMS7000.COM, using the previously linked program PRG as an argument.

   ```
   DMS7000 PRG
   ```

If the alternative installation method described in Appendix D is used, then the control procedure described in Appendix D applies.

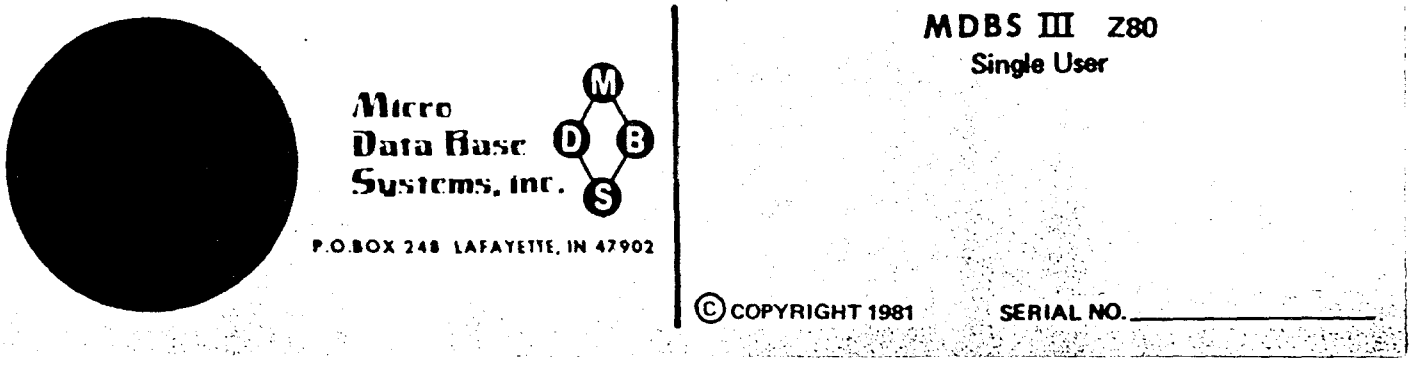
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



VII. DML COMMAND FORM

BASCOM is a block oriented language that permits direct or indirect invocation of DML commands (see the block/direct or block/indirect example for each DML command in the MDBS DMS Manual). The precise calling forms for both direct and indirect DML usage are presented in Appendix A and are illustrated in the examples below. Exclusive use of the direct approach is recommended, since it is easier to use. The indirect is provided solely for compatability with interpretive BASIC (MBASIC), which does not support direct invocation. **There is no guarantee of continuing support for the indirect approach.**

A. Command Status

The command status variable must be an integer variable.

B. Special Commands

The VARCS, VARCMD, SETPBF and DEFINE commands are required in this environment. The EXTEND, UNDEF and ALTEOS commands can also be used.

The **first** DML command used in any application program must be VARCS. For instance, if the command status variable is named E0%, then the first DML command is

```
CALL VARCS (E0%)
```

The **second** DML command used in any application program must be VARCMD. This command is peculiar to MBASIC and BASCOM. Its argument is a host language string variable (e.g., CMD$). When a DML command (other than VARCS, VARCMD, DEFINE, EXTEND, UNDEFINE, and SETPBF) is invoked, it uses the value of CMD$ as its command string. For instance, to declare the command string variable CMD$:

```
CALL VARCMD (CMD$)
```

As an example of SETPBF, 4096 bytes can be allocated for program buffers as follows:

```
DIM DMSBUFFER%(2047)
I% = 4096
CALL SETPBF (DMSBUFFER%(0),I%)
```

Since integers are two bytes and DMSBUFFER% subscripting begins with 0, the dimension of DMSBUFFER% is declared to be half of the desired buffer size minus 1.

Suppose that we want to create a data block called BLK, consisting of the two variables VAR1% and VAR2%. The following examples show how to accomplish this in the direct and indirect forms.

DIRECT

```
X$="BLK":X%=2
CALL DEFINE(X$,X%,VAR1%,VAR2%)
```

INDIRECT

```
X$="DEFINE,BLK":X%=2
CALL DMSD(E0%,X$,X%,VAR1%,VAR2%)
```

To extend BLK to include another variable (VAR3%):

DIRECT

```
X$="BLK":X%=1
CALL EXTEND(X$,X%,VAR3%)
```

INDIRECT

```
X$="EXTEND,BLK":X%=1
CALL DMSD(E0%,X$,X%,VAR3%)
```

To undefine all data blocks:

DIRECT

```
CALL UNDEF
```

INDIRECT

```
X$="UNDEF"
CALL DMSD(E0%,X$)
```

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.

DIRECT

```
CALL ALTEOS
```

INDIRECT

```
CMD$="ALTEOS"
CALL DMS
```

C. Open and Close Command Examples

DIRECT

```
X$="OBLK":X%=4
CALL DEFINE(X$,X%,DBU$,DBP$,
   DBO$,DBF$)
  .
  .
  .
CMD$="OBLK"
CALL DBOPN
  .
  .
  .
CALL DBCLS
```

INDIRECT

```
X$="DEFINE,OBLK":X%=4
CALL DMSD(E0%,X$,X%,DBU$,DBP$,
   DBO$,DBF$)
  .
  .
  .
CMD$="DBOPN,OBLK"
CALL DMS
  .
  .
  .
CMD$="DBCLS"
CALL DMS
```


D. Find Command Examples

DIRECT

```
X$="BLK":X%=1
CALL DEFINE(X$,X%,VAR%)
 .
 .
 .
CMD$="SET1,BLK"
CALL FMSK
```

INDIRECT

```
X$="DEFINE,BLK":X%=1
CALL DMSD(E0%,X$,X%,VAR%)
 .
 .
 .
CMD$="FMSK,SET1,BLK"
CALL DMS
```

Here, SET1 is the name of a set that has been specified in a DDL specification.

E. Get and Put Command Examples

DIRECT

```
X$="BLK":X%=1
CALL DEFINE(X$,X%,VAR!)
 .
 .
 .
CMD$="YTDEARN,BLK"
CALL GFC
VAR!=VAR!+1029.00
CALL PFC
```

INDIRECT

```
X$="DEFINE,BLK":X%=1
CALL DMSD(E0%,X$,X%,VAR!)
 .
 .
 .
CMD$="GFC,YTDEARN,BLK"
CALL DMS
VAR!=VAR!+1029.00
CMD$="PFC,YTDEARN,BLK"
CALL DMS
```

IMPORTANT NOTE: When using Get or Put commands, care should be taken to initialize any program variable that corresponds to a character data item. The length of each such variable should be initialized to the maximum length declared in the DDL specification for the corresponding data item. Failure to observe this rule yields a command status error of 33.

F. Assignment Command Examples

DIRECT

```
X$="BLK":X%=1
CALL DEFINE(X$,X%,VAR%)
 .
 .
 .
VAR%=3
CMD$="SET1,BLK"
CALL SMU
 .
 .
 .
CMD$="SET1,SET2"
CALL SOM
```

INDIRECT

```
X$="DEFINE,BLK":X%=1
CALL DMSD(E0%,X$,X%,VAR%)
 .
 .
 .
VAR%=3
CMD$="SMU,SET1,BLK"
CALL DMS
 .
 .
 .
CMD$="SET1,SET2"
CALL DMS
```

Here, SET1 and SET2 are names of sets that have been specified in a DDL specification.

G. Utility Command Examples

DIRECT

```
X$="BLK1":Y$="BLK2":X%=1
CALL DEFINE(X$,X%,VAR%)
CALL DEFINE(Y$,X%,VAR$)
 .
 .
 .
CMD$="SET1,BLK1"
CALL GMC
 .
 .
 .
CMD$="SET3,BLK2"
CALL GTM
 .
 .
 .
CMD$="SALEMP,POSSESS"
CALL TOT
```

INDIRECT

```
X$="DEFINE,BLK1"
Y$="DEFINE,BLK2":X%=1
CALL DMSD(E0%,X$,X%,VAR%)
CALL DMSD(E0%,Y$,X%,VAR$)
 .
 .
 .
CMD$="GMC,SET1,BLK1"
CALL DMS
 .
 .
 .
CMD$="GTM,SET3,BLK2"
CALL DMS
 .
 .
 .
CMD$="TOT,SALEMP,POSSESS"
CALL DMS
```

Here, SET1, SET3 and POSSESS are names of sets that have been specified in a DDL specification. As shown in Table V-2, VAR% must be integer (i.e., consistent with a 4 byte unsigned data item). Notice that RNAM$ is character, since GTM requires a variable consistent with an 8 byte character data item. The record type name returned by GTM will be in upper case, with blank fill (this same convention also holds for GTO, GTC).

H. Chaining

Immediately prior to chaining, DBSAVE should be invoked.

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, IDML6.OVL, IDML7.OVL, IDML8.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.

If IDML has been configured with MDBS-RTL, a log file name (fully qualified) can be optionally specified on the command line with a -R parameter. For instance, to execute IDML with a log file named LOG.SUB,

```
IDML START.IDM -B2560 -RLOG.SUB
```

would be used. If a log file name is omitted from a command line, the log file name in the DDL specification is assumed.

When switching floppy disks to change on-line areas, exit from IDML, switch the disks, press control-C and re-enter IDML.

C. MDBS-QRS

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

QRS.COM, QRS0.OVL, QRS1.OVL, QRS2.OVL, QRS3.OVL, QRS4.OVL, QRS5.OVL, QRS6.OVL, QRS7.OVL, QRS8.OVL, QRS9.OVL, QRS10.OVL

Omitting QRS3.OVL has the effect of disabling the DEFINE command.

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

```
QRS
```

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.QRS and allocate 2560 bytes, the operating system command line is:

```
QRS START.QRS  -B2560
```

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

The special format generated by the SPEW command in this environment is the .CAL file format of SuperCalc (version 1.07 and greater). When SPEW is used, the file name specified with the FN parameter should have the .CAL extension so that it can be immediately used as input to SuperCalc.

A fully qualified file name must be used with the QRS READ command and with the FN parameter (see I-B). When the READ command is used, the file name must be in quotes (either single or double).

When switching floppy disks to change on-line areas, exit QRS, swich disks, press control-C, and re-enter QRS.

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.

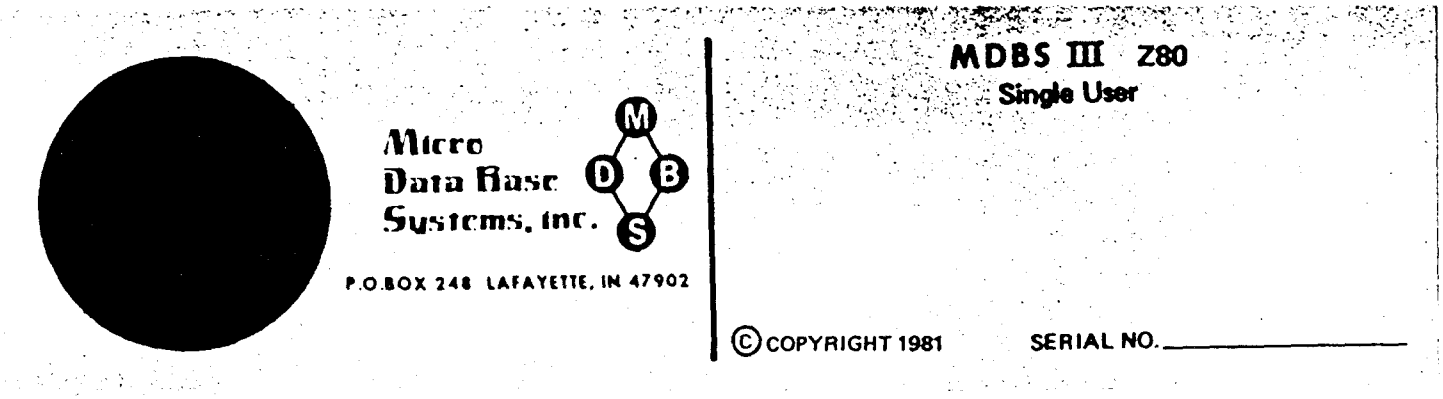
This page intentionally left blank.

Do not purchase or otherwise accept MDBS III software products not bearing the official colored MDBS diskette/tape label. The serial number on the label uniquely identifies the system licensee. If you are approached by a person or organization attempting to illegally distribute MDBS products, please contact

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616 TWX810-342-1881

We appreciate your assistance in preventing software piracy.

Sample of MDBS III diskette/tape label:



Appendix A

DIRECT INVOCATION

```
CALL ALTEOS

CMD$="SET1,SET2,SET3"
CALL AMM

CMD$="SET1,SET2,SET3"
CALL AMO

CMD$="SET1,SET2,SET3"
CALL AOM

CMD$="SET1,SET2,SET3"
CALL AOO

CMD$="BLK"
CALL AUI

CMD$="BLK"
CALL CCU

CMD$="RECORD,AREA,BLK"
CALL CRA

CMD$="RECORD,BLK"
CALL CRS

CALL DBCLS

CMD$="AREA"
CALL DBCLSA

CMD$="BLK"
CALL DBENV

CMD$="BLK"
CALL DBOPN

CMD$="AREA,BLK"
CALL DBOPNA

CALL DBSAVE

CMD$="BLK"
CALL DBSTAT

X$="BLK":X%=n
CALL DEFINE(X$,X%,vl,...,vn)

CALL DRC

CMD$="SET"
CALL DRM

CMD$="SET"
CALL DRO

X$="BLK":X%=n
CALL EXTEND(X$,X%,vl,...,vn)

CMD$="RECORD,BLK"
CALL FDRK

CMD$="SET"
CALL FFM

CMD$="SET"
CALL FFO

CMD$="AREA"
CALl FFS

CMD$="SET"
CALL FLM

CMD$="SET"
CALL FLO

CMD$="ITEM,SET,BLK"
CALL FMI

CMD$="SET,BLK"
CALL FMSK

CMD$="SET"
CALL FNM

CMD$="ITEM,SET,BLK"
CALL FNMI

CMD$="SET,BLK"
CALL FNMSK

CMD$="SET"
CALL FNO

CMD$="ITEM,SET,BLK"
CALL FNOI

CMD$="SET,BLK"
CALL FNOSK
```



```
CMD$="AREA"
CALL FNS

CMD$="ITEM,SET,BLK"
CALL FOI

CMD$="SET,BLK"
CALL FOSK

CMD$="SET"
CALL FPM

CMD$="SET"
CALL FPO

CMD$="RECORD,BLK"
CALL FRK

CMD$="BLK"
CALL GETC

CMD$="SET,BLK"
CALL GETM

CMD$="SET,BLK"
CALL GETO

CMD$="ITEM,BLK"
CALL GFC

CMD$="ITEM,SET,BLK"
CALL GFM

CMD$="ITEM,SET,BLK"
CALL GFO

CMD$="SET,BLK"
CALL GMC

CMD$="SET,BLK"
CALL GOC

CMD$="BLK"
CALL GTC

CMD$="SET,BLK"
CALL GTM

CMD$="SET,BLK"
CALL GTO

CMD$="SET"
CALL IMS

CMD$="SET"
CALL IOS

CALL LGCPLX

CALL LGENDX

CMD$="BLK"
CALL LGFILE

CALL LGFLSH

CMD$="BLK"
CALL LGMSG

CMD$="BLK"
CALL MAU

CMD$="BLK"
CALL MCC

CALL MCF

CALL MCP

CMD$="RECORD"
CALL MRTF

CMD$="RECORD"
CALL MRTP

CMD$="SET"
CALL MSF

CMD$="SET"
CALL MSP

CALL NCI

CMD$="ITEM,BLK"
CALL PFC

CMD$="ITEM,SET,BLK"
CALL PFM

CMD$="ITEM,SET,BLK"
CALL PFO

CMD$="BLK"
CALL PIFD

CMD$="BLK"
CALL PUTC
```

```
CMD$="SET,BLK"
CALL PUTM

CMD$="SET,BLK"
CALL PUTO

CMD$="SET"
CALL RMS

CMD$="SET"
CALL ROS

CMD$="SET"
CALL RSM

CMD$="SET"
CALL RSO

CMD$="SET"
CALL SCM

CALL SCN

CMD$="SET"
CALL SCO

CMD$="BLK"
CALL SCU

I%=bufsize
CALL SETPBF(BUF(Ø),I%)

CMD$="SET"
CALL SMC

CMD$="SET"
CALL SME

CMD$="SET1,SET2"
CALL SMM

CMD$="SET"
CALL SMN

CMD$="SET1,SET2"
CALL SMO

CMD$="SET,BLK"
CALL SMU

CMD$="SET"
CALL SOC

CMD$="SET"
CALL SOE

CMD$="SET1,SET2"
CALL SOM

CMD$="SET"
CALL SON

CMD$="SET1,SET2"
CALL SOO

CMD$="SET,BLK"
CALL SOU

CMD$="BLK"
CALL SUC

CMD$="SET,BLK"
CALL SUM

CMD$="BLK"
CALL SUN

CMD$="SET,BLK"
CALL SUO

CMD$="BLK"
CALL SUU

CMD$="RECORD"
CALL TCT

CMD$="RECORD,SET"
CALL TMT

CMD$="RECORD,SET"
CALL TOT

CALL TRABT

CALL TRBGN

CALL TRCOM

CALL UNDEF

CALL VARCS(E0%)

CALL VARCMD(CMD$)

CMD$="SET1,SET2,SET3"
CALL XMM

CMD$="SET1,SET2,SET3"
CALL XMO

CMD$="SET1,SET2,SET3"
CALL XOM

CMD$="SET1,SET2,SET3"
CALL XOO
```



```
CMD$="ALTEOS"
CALL DMS

CMD$="AMM,SET1,SET2,SET3"
CALL DMS

CMD$="AMO,SET1,SET2,SET3"
CALL DMS

CMD$="AOM,SET1,SET2,SET3"
CALL DMS

CMD$="AOO,SET1,SET2,SET3"
CALL DMS

CMD$="AUI,BLK"
CALL DMS

CMD$="CCU,BLK"
CALL DMS

CMD$="CRA,RECORD,AREA,BLK"
CALL DMS

CMD$="CRS,RECORD,BLK"
CALL DMS

CMD$="DBCLS"
CALL DMS

CMD$="DBCLSA,AREA"
CALL DMS

CMD$="DBENV,BLK"
CALL DMS

CMD$="DBOPN,BLK"
CALL DMS

CMD$="DBOPNA,AREA,BLK"
CALL DMS

CMD$="DBSAVE"
CALL DMS

CMD$="DBSTAT,BLK"
CALL DMS

X$="DEFINE,BLK":X%=n
CALL DMSD(E0%,X$,X%,vl,...,vn)

CMD$="DRM,SET"
CALL DMS

CMD$="DRO,SET"
CALL DMS

X$=EXTEND,BLK":X%=n
CALL DMSD(E0%,X$,X%,vl,...,vn)

CMD$="FDRK,RECORD,BLK"
CALL DMS

CMD$="FFM,SET"
CALL DMS

CMD$="FFO,SET"
CALL DMS

CMD$="FFS,AREA"
CALL DMS

CMD$="FLM,SET"
CALL DMS

CMD$="FLO,SET"
CALL DMS

CMD$="FMI,ITEM,SET,BLK"
CALL DMS

CMD$="FMSK,SET,BLK"
CALL DMS

CMD$="FNM,SET"
CALL DMS

CMD$="FNMI,ITEM,SET,BLK"
CALL DMS

CMD$="FNMSK,SET,BLK"
CALL DMS

CMD$="FNO,SET"
CALL DMS

CMD$="FNOI,ITEM,SET,BLK"
CALL DMS

CMD$="FNOSK,SET,BLK"
CALL DMS

CMD$="FNS,AREA"
CALL DMS

CMD$="FOI,ITEM,SET,BLK"
CALL DMS
```



```
CMD$="FOSK,SET,BLK"
CALL DMS

CMD$="FPM,SET"
CALL DMS

CMD$="FPO,SET"
CALL DMS

CMD$="FRK,RECORD,BLK"
CALL DMS

CMD$="GETC,BLK"
CALL DMS

CMD$="GETM,SET,BLK"
CALL DMS

CMD$="GETO,SET,BLK"
CALL DMS

CMD$="GFC,ITEM,BLK"
CALL DMS

CMD$="GFM,ITEM,SET,BLK"
CALL DMS

CMD$="GFO,ITEM,SET,BLK"
CALL DMS

CMD$="GMC,SET,BLK"
CALL DMS

CMD$="GOC,SET,BLK"
CALL DMS

CMD$="GTC,BLK"
CALL DMS

CMD$="GTM,SET,BLK"
CALL DMS

CMD$="GTO,SET,BLK"
CALL DMS

CMD$="IMS,SET"
CALL DMS

CMD$="IOS,SET"
CALL DMS

CMD$="LGCPLX"
CALL DMS

CMD$="LGENDX"
CALL DMS

CMD$="LGFILE,BLK"
CALL DMS

CMD$="LGFLSH"
CALL DMS

CMD$="LGMSG,BLK"
CALL DMS

CMD$="MAU,BLK"
CALL DMS

CMD$="MCC,BLK"
CALL DMS

CMD$="MCF"
CALL DMS

CMD$="MCP"
CALL DMS

CMD$="MRTF,RECORD"
CALL DMS

CMD$="MRTP,RECORD"
CALL DMS

CMD$="MSF,SET"
CALL DMS

CMD$="MSP,SET"
CALL DMS

CMD$="NCI"
CALL DMS

CMD$="PFC,ITEM,BLK"
CALL DMS

CMD$="PFM,ITEM,SET,BLK"
CALL DMS

CMD$="PFO,ITEM,SET,BLK"
CALL DMS

CMD$="PIFD,BLK"
CALL DMS

CMD$="PUTC,BLK"
CALL DMS

CMD$="PUTM,SET,BLK"
CALL DMS

CMD$="PUTO,SET,BLK"
CALL DMS
```



```
CMD$="RMS,SET"
CALL DMS

CMD$="ROS,SET"
CALL DMS

CMD$="RSM,SET"
CALL DMS

CMD$="RSO,SET"
CALL DMS

CMD$="SCM,SET"
CALL DMS

CMD$="SCN"
CALL DMS

CMD$="SCO,SET"
CALL DMS

CMD$="SCU,BLK"
CALL DMS

I%=bufsize
CALL SETPBF(BUF(Ø),I%)

CMD$="SMC,SET"
CALL DMS

CMD$="SME,SET"
CALL DMS

CMD$="SMM,SET1,SET2"
CALL DMS

CMD$="SMN,SET"
CALL DMS

CMD$="SMO,SET1,SET2"
CALL DMS

CMD$="SMU,SET,BLK"
CALL DMS

CMD$="SOC,SET"
CALL DMS

CMD$="SOE,SET"
CALL DMS

CMD$="SOM,SET1,SET2"
CALL DMS

CMD$="SON,SET"
CALL DMS

CMD$="SOO,SET1,SET2"
CALL DMS

CMD$="SOU,SET,BLK"
CALL DMS

CMD$="SUC,BLK"
CALL DMS

CMD$="SUM,SET,BLK"
CALL DMS

CMD$="SUN,BLK"
CALL DMS

CMD$="SUO,SET,BLK"
CALL DMS

CMD$="SUU,BLK"
CALL DMS

CMD$="TCT,RECORD"
CALL DMS

CMD$="TMT,RECORD,SET"
CALL DMS

CMD$="TOT,RECORD,SET"
CALL DMS

CMD$="TRABT"
CALL DMS

CMD$="TRBGN"
CALL DMS

CMD$="TRCOM"
CALL DMS

$X="UNDEF"
CALL DMSD

CALL VARCS(EØ%)

CALL VARCMD(CMD$)

CMD$="XMM,SET1,SET2,SET3"
CALL DMS

CMD$="XMO,SET1,SET2,SET3"
CALL DMS

CMD$="XOM,SET1,SET2,SET3"
CALL DMS

CMD$="XOO,SET1,SET2,SET3"
CALL DMS
```

APPENDIX B

DML Retrieval/Write Command Groups

Appendix B

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	tct
ccu	ffm	fnoi	gfm	scu	soo	tmt
cmt	ffo	fnosk	gfo	setpbf	sor	toggle
cot	ffs	fns	gfr	smc	sou	tot
dbcls	findm	foi	gmc	sme	src	undef
dbclsa	findo	fosk	goc	smm	srm	varcmd
dbenv	flm	fpm	gtc	smn	srn	varcs
dbopn	flo	fpo	gtm	smo	sro	
dbopna	fmi	frk	gto	smr	suc	
dbsave	fmsk	getc	nci	smu	sum	
dbstat	fnm	getm	scm	soc	sun	

R1W1

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

R2W1

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

R3W1

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

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

R1W3

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

R2W3

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

R3W3

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

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

APPENDIX C

First Alternative MDBS DMS Installation Method

Appendix C

This alternative installation method allows the use of chaining within application programs. If chained programs are used with a DMS that has been installed as described here, then DBSAVE must be the last command invoked prior to chaining and any variables used in the VARCS, VARCMD, DEFINE, EXTEND, and SETPBF commands must be in COMMON.

A. Installation

Copy MKDMS, MKTAB, the MLINK files, DBRUN.REL, DMS.REL, OSCPM.REL, A.REL, Z.REL, and BASCOM80.REL to the same working disk.

Now execute MKDMS. This interactive installation program generates a command file which will create a DMS runtime module that supports desired DML commands. Appendix B 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 A. Operator responses are shown in bold face type.

```
MKDMS v1.01
(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
decypherable).  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.

```
Do you have an RTL form of MDBS (y/n) [n]?
Do you have a MULTI USER system (y/n) [n]?
Do you want FAST I/O (y/n) [n]?
Do you want to disable CALC processing (y/n) [n]?
Do you want to disable DATE processing (y/n) [n]?
Do you want to disable FLOAT processing (y/n) [n]?
Do you want to disable IDEC processing (y/n) [n]?
Do you want to disable REAL processing (y/n) [n]?
Do you want to disable TIME processing (y/n) [n]?
```

Next, select the language interface you intend to use. The languages supported are:

```
1) BASCOM
2) CB80
3) COBOL80
4) FORTRAN80
5) PASCALMT+
6) PASCALZ
7) ASM
8) C'
```

Enter desired language: **BASCOM**

Now, you must select which DMS commands you need to be supported by the runtime module. This is accomplished by selecting one of the 15 predefined groupings that are listed in your system specific manual or by selecting each DML command separately as it is displayed. The groupings are **RnWm where n** is a digit from 1 to 3 and m is a digit from 0 to **4**. If you want to select your own group of DML commands, enter an 'S'. For convenience, if you decide you need all DML commands, enter an 'A'.

Enter the desired command level (RnWm/A/S) [S]: **R2W1**

Next, the runtime module's output file (fn) and/or drive must be specified. The output file name will be used for the four different files created by MKDMS. A single name (fn) is used to keep track of which set of files belongs to which runtime module. These four files will be named:

```
fn.MLA  = list of commands for MLINK
fn.COM  = dms runtime produced by MLINK
fn.REL  = file produced by MKTAB of dms command locations
fn.SYM  = symbol  table produced by MLINK and used as  input
          to MKTAB
```

Remember, just enter a name and/or a drive. The proper extensions are automatically appended by MKDMS.

Enter output file name and/or drive [DMS7000]: **a:**

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, OSCPM.REL, A.REL, Z.REL, and BASCOM80.REL.

These files will be used to construct your DMS runtime routine 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 (a:DMS7000.SUB) which contains the commands that will create your DMS runtime module. To execute this command file, enter the following command:
SUBMIT a:DMS7000.SUB

Now, to create the DMS runtime module, execute the command file created by MKDMS. This command file invokes the MLINK and MKTAB programs. For example:

SUBMIT A:DMS7000.SUB

```
MDBS MLINK V1.02
(C) COPYRIGHT 1982, Micro Data Base Systems, Inc.
Lafayette, IN  47902

DBRUN:
     DBRUN
1 modules

BASCOM80:
     DDMDBS ALTEOS VARCMD VARCS  LNBCOM GETFWA EIINT  IEINT
     EIREAL IEREAL EISTR  IESTR  EICHAR IECHAR
14 modules

DMS:
     CRS    FDRK   F2MDBS FRK    IMS    RCMDBS DBSTAT DEFINE
     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
68 modules

OSTRS:
     DFCL   DFSK   DFTE   DFOPN  DFRW   DFGN   DFPN   FCB
8 modules

A:
     DIVZ8  DIVZ16 MULZ16 MULZ8
4 modules
```

```
Z:
     ZZZLWA
1 modules
Code = 9280 Common = E904 Data = EBB2
Xfer = 9280 Next = EFDB

DBRUN:
     DBRUN

BASCOM80:
     DDMDBS ALTEOS VARCMD VARCS  LNBCOM GETFWA EIINT  IEINT
     EIREAL IEREAL EISTR  IESTR  EICHAR IECHAR

DMS:
     CRS    FDRK   F2MDBS FRK    IMS    RCMDBS DBSTAT DEFINE
     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

OSTRS:
     DFCL   DFSK   DFTE   DFOPN  DFRW   DFGN   DFPN   FCB

A:
     DIVZ8  DIVZ16 MULZ16 MULZ8

Z:
     ZZZLWA
Linking complete.

MKTAB v1.01 [82/03/26]
COPYRIGHT (C) 1982, Micro Data Base Systems, Inc.

Reading directive file DMS7000/MLA:1
Reading symbol table DMS7000:1
Writing rel file DMS7000:1
```

The installation of DMS is now complete.

B. Testing

After the installation of MDBS.DDL and MDBS.DMS, the sample DDL specification and sample application program can be used to test the success of the installation procedure. The sample application requires a minimal command grouping of R2W1.

1. To initialize the sample data base, enter:

```
DDL SAMPLE
```

2. Compile the sample application program by entering:

```
BASCOM =SAMPLE/E/C
```

3. Resolve unsatisfied externals as follows:

```
L80 SAMPLE,DMS7000,SAMPLE/N/E
```

4. Run the sample application program by executing DMS7000, using SAMPLE as an argument.

```
DMS7000 SAMPLE
```


APPENDIX D

Second Alternative Installation Method

Appendix D

This appendix describes how to use MDBS in conjunction with chained BASCOM programs (with BASCOM'S BRUN option).

Prepare your application programs (using direct DML invocation) as described in Chapter VII of this manual, with the following exceptions:

1. Before invoking VARCS in the first of the chained modules, insert the following call:

   ```
   CALL DBINIT
   ```

   Note that DBINIT should never be called while the data base is open.

2. Any variable that is used with a DEFINE or EXTEND command must have been declared in a COMMON statement.

3. The array used with SETPBF must have been declared in a COMMON statement.

4. The first two DML commands of any chained module (containing DML commands) must be:

   ```
   CALL VARCS(E0%)
   CALL VARCMD(CMD$)
   ```

5. The last DML command of any chained module (containing DML commands) must be:

   ```
   CALL DBCHN
   ```

   DBCHN automatically performs a DBSAVE. (Exception: In the multiuser case, DBCHN is not used. Instead, the data base must be closed before chaining and then reopened after VARCMD in the chained module.)

The control procedure for compiling, linking and executing a chained application program is as follows:

1. Before linking, use the DATOCOD utility to prepare new relocatable files for use in the linking process. You choose the new file names. For example, DMS.REL becomes DMSC.REL. DATOCOD transfers MDBS constants from the data areas to the code areas. This is accomplished by entering the following operating system command lines:

   ```
                                DATOCOD DMS.REL DMSC.REL
   (or for RTL form of DMS:     DATOCOD RTL.REL RTLC.REL)
                                DATOCOD BASCOM80.REL BASCOMC.REL
                                DATOCOD A.REL AC.REL
                                DATOCOD OSCPM.REL OSCPMC.REL
                                DATOCOD Z.REL ZC.REL
   ```


If you have a library manager such a LIB or LIB80, create CPMDMS.REL by entering the following operating system command line:

```
LIB CPMDMS=DMSC,OSCPMC,AC,ZC
```

This assumes that the DMSC.REL, OSCPMC.REL, AC.REL, and ZC.REL files are on the default drive. If you intend to use the RTL form of MDBS, the same approach is employed, except use RTLC instead of DMSC and CPMRTL instead of CPMDMS.

2. Compile the program as usual without the /O option.

3. Use MLINK with -K0 as the first argument. **If you used the library manager** in step 1, then enter:

```
MLINK -K0 PRG COMDEF BASCOMC -LCPMDMS
```

where PRG is the object program and where COMDEF.REL, BASLIB.REL, and BCLOAD (the latter two are supplied with BASCOM) are on the default drive at link time. For the RTL form of MDBS, use -LCPMRTL instead of -LCPMDMS.

If you did not use a library manager in step 1, then use the following MLINK command line:

```
MLINK -K0 PRG COMDEF BASCOMC -LDMSC -LOSCPMC -LAC -LZC
```

where PRG is on the object program and where COMDEF.REL, BASLIB.REL and BCLOAD are on the default drive at link time. For the RTL form of MDBS, use -LRTLC instead of -LDMSC.

The -K0 option used with MLINK prevents the initialization of initialized common blocks, so that chaining can occur. Ignore the MLINK error message "common block // size mismatch" when running MLINK.

4. Repeat steps 2 and 3 for each of the other programs involved in the chaining. COMDEF must be linked, even for a chained program containing no DML commands.

5. Execute the linked program:

```
PRG
```

To test this procedure, five files containing chained modules are provided SAMPLEC1.BAS, ..., SAMPLEC5.BAS (Exception: SAMPLEC5 is not provided with multiuser versions of MDBS). Use steps 2 and 3 above for each of these five files. Then execute the program by entering:

```
SAMPLEC1
```