SYS.MBASIC

MDBS - CP/M Interface Notes

CP/M with Microsoft BASIC Interpreters

```
mm        mm  dddddddd    bbbbbbbb       ssssss
mmm      mmm  ddddddddd   bbbbbbbbb    ssssssss
mmmm   mmmm  dd       dd  bb      bb  ss        ss
mm mmmm mm  dd       dd  bb      bb  ss
mm   mm   mm  dd       dd  bbbbbbbbb   sssssssss
mm        mm  dd       dd  bbbbbbbbb    sssssssss
mm        mm  dd       dd  bb      bb             ss
mm        mm  dd       dd  bb      bb  ss        ss
mm        mm  ddddddddd   bbbbbbbbb    ssssssss
mm        mm  dddddddd    bbbbbbbb       ssssss
```

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

(317) 448-1616

Mahan

8:30

September 1980

OUTLINE

General Notes

For

Installing and Using MDBS with a Host Language Interpreter

These general notes provide an outline on how to install the MDBS software on your system. The system specific portion of this manual discusses the details of preforming the following for your configuration. The general stages are:

First: Installing and using MDBS on your system (Figure A).

As a result of this installation process you will have two executable files:

(a) An executable form of MDBS.DDL

(b) An executable form of MDBS.DMS that has been attached to your host language interpreter.[1]

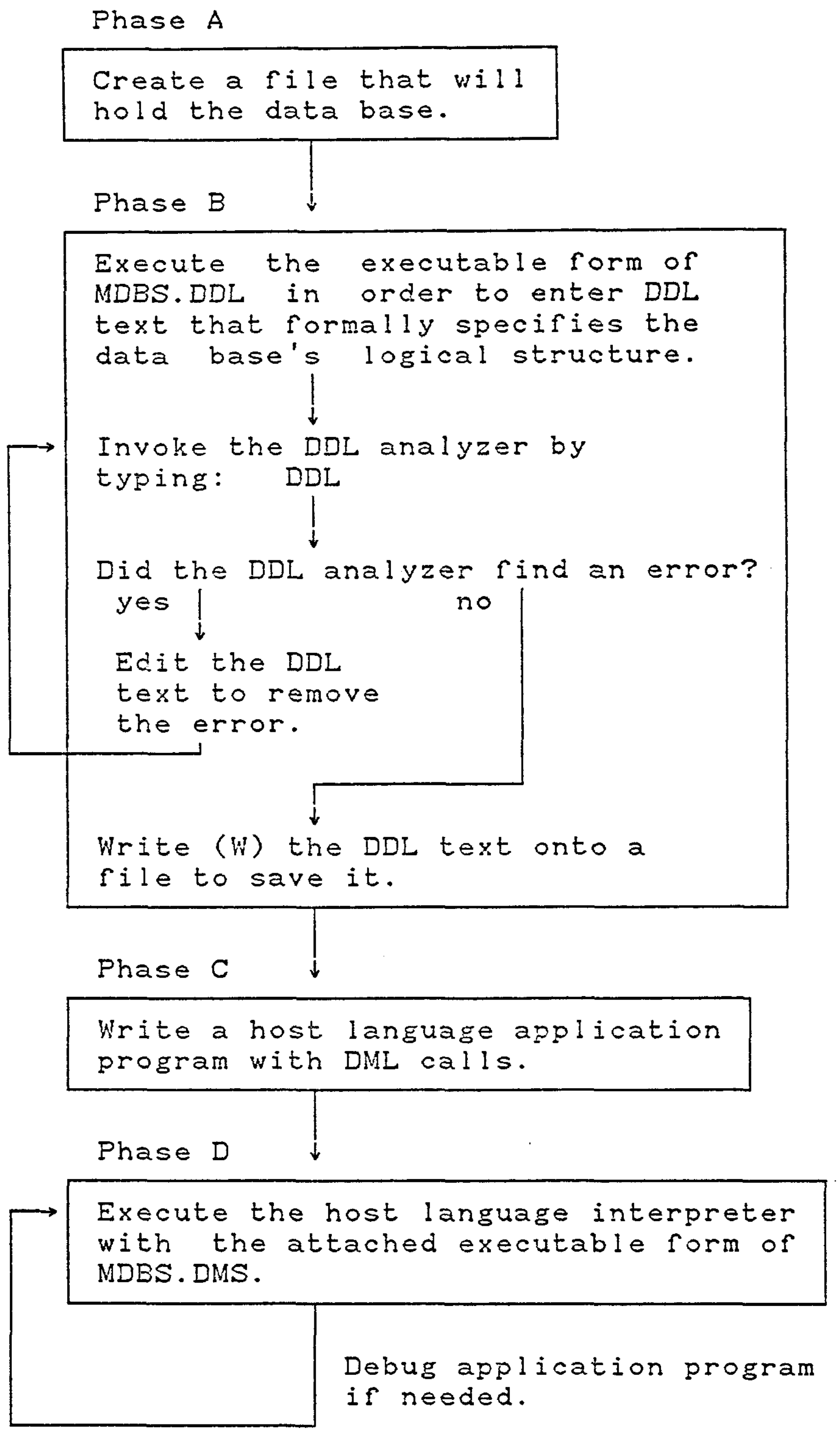
Second: Use MDBS to implement an application system (Figure B).

The executable form of MDBS.DDL is used to define a data base. Application programs that access the data base are interpreted by your host language interpreter that has an executable form of MDBS.DMS attached to it.[1]

[1]Under the TRSDOS (Models I and II), the NEWDOS and the Apple DOS operating systems, the executable form of MDBS.DMS is not attached to the host language interpreter. Both the interpreter and the executable form of MDBS.DMS are in memory during application program execution.

Convention: In all walk-through examples, the underlined portion is what you type in; the non-underlined portion is displayed by the computer.

The specifics of performing the first stage are provided below.



Phase A
Create a file that will hold the data base.
Phase B
Execute the executable form of MDBS.DDL in order to enter DDL text that formally specifies the data base's logical structure.
Invoke the DDL analyzer by typing: DDL
Did the DDL analyzer find an error?
yes
no
Edit the DDL text to remove the error.
Write (W) the DDL text onto a file to save it.
Phase C
Write a host language application program with DML calls.
Phase D
Execute the host language interpreter with the attached executable form of MDBS.DMS.
Debug application program if needed.

Figure B. Using MDBS to Implement an Application System

Phases A-D: Details

Phase A:

Create a file that will hold the data base. This file must have the same name as given on the FILE line in the DDL description. This file must be created on drive 1 (and on any other drives specified with the DRIVE card(s) in the DDL).

Phase B:

Execute the executable form of MDBS.DDL that was supplied (or which you have created in Phase 1). Now you are within the MDBS DDL system, so you can:

(a) Read a file of DDL source text (e.g.,the files INVNTRY and SAMPLDDL supplied on the diskettes you received).

(b) Generate a file of DDL source text for your own data base application (text entry aids are provided).

(c) Edit a file of DDL source text.

(d) Write (i.e., save on disk) a file of DDL source text.

(e) Submit a file of DDL source text to the DDL analyzer which checks your DDL text for correctness. Diagnostics are issued for errors; otherwise the message DDL PROCESSING COMPLETE is returned, indicating that the data base described in the DDL source text is now defined for the file created in Phase A. This data base file now contains a data base directory composed of all schema information.

Commands to accomplish the foregoing kinds of activities (along with descriptions of error messages returned from the DDL analyzer) are described in the MDBS User's Manual.

Phase C:

Write host language application programs (containing MDBS DML calls) for a data base defined in Phase B(e). These programs are used to add data to the data base, extract data from the data base, change data and their relationships within the data base, and/or delete data from the data base. (The SAMPLE file on the diskettes you received is an example application program for a data base whose source DDL text exists on the SAMPLDDL file.) The method for calling DML commands from a host language depends upon the nature of the host language. The necessary calling protocol for your host language is illustrated in the MDBS system specific manual.

The nature of each MDBS DML command that can be called from your host language is described in the MDBS User's Manual. Errors that can result from the incorrect DML usage are described for each command. It is suggested that you follow through the code of SAMPLE in order to become oriented to DML applications programming with your host language.

Phase D:

Execute the host language interpreter having the attached executable form of MDBS.DMS (generated in Phase 4)[1]. Load and run an application program devised in Phase C.

This completes the General Notes section. We now turn to the specifics of running MDBS with your configuration.

[1]Under TRSDOS(I and II), NEWDOS and Apple DOS the executable form of MDBS.DMS is not attached to the host language interpreter.

I. INTRODUCTION

The MDBS.DDL and MDBS.DMS User's Manual was written in a non-machine-dependent manner. The purpose of the manual is to discuss the specifics of running the MDBS packages on CP/M systems with the MICROSOFT BASIC-80 interpreter.

The following files are on the MDBS diskette(s):

1. DDL.COM - An absolute version of MDBS.DDL loading at 100 hex
2. INVNTRY.DDL - A sample data description file
3. SAMPLDDL.DDL - Sample DDL file for use with BASSAM.TXT
4. RLC.COM - An absolute relocator
5. RLC.IHF - A relocatable relocator
6. DDL.IHF - A relocatable version of the MDBS.DDL system
7. SAMPLE.BAS - Sample BASIC Program
8. DMS.COM - An absolute version of MDBS.DMS loading at 7400 hex
9. DMS.IHF - A relocatable version of the MDBS.DMS system

Section II gives a step-by-step walk through of Phases 0-4 for installing MDBS and Phases A-D for using MDBS on CP/M systems with MICROSOFT BASIC-80 (MBASIC).

MDBS.DDL is your personalized Data Definition Language Analyzer/ Editor and MDBS.DMS your Data Management System. An executable form of MDBS.DDL is provided on the DDL.COM file which loads at 100H. Section II (Phase I) shows how to ORG the relocatable form of MDBS.DDL (on file DDL.IHF) to 100H. Both DDL.COM and DDL.IHF use standard CP/M

I/O entry points. If you are using non standard I/O entries or require other patching, refer to Section IV for patching procedures.

DMS.COM contains an executable form of MDBS.DMS that loads at 7400H.[1] DMS.IHF contains the relocatable form of MDBS.DMS. In order to ORG it to 7400 (or some other address) follow the procedure given in section II (Phase II). If you are using nonstandard CP/M procedures, refer to Section V for patching procedures.

In Section VI we show how to call the DMS routines from MICROSOFT BASIC 80 versions 4.xx and 5.xx.

Finally, in Section VII we provide some important miscellaneous notes for the CP/M user of the MDBS package.

You can run your version of DDL.COM by following the procedure given in Figure 1. At the end of Step 2 you are in the DDL program. Refer to Section II.E of the MDBS User's Manual for the DDL commands. However, to quickly see some action, continue with the procedure in Figure 1.

Steps 3-5 result in the reading of a sample data base description from the file SAMPLDDL.DDL. Step 6 results in the listing of the sample description. Step 8 returns control to the operating system.

[1] The size of the DMS.IHF system will vary depending on the functions available and the operating system/host language selected. Use the RLC routine on the relocatable forms to determine the exact length.

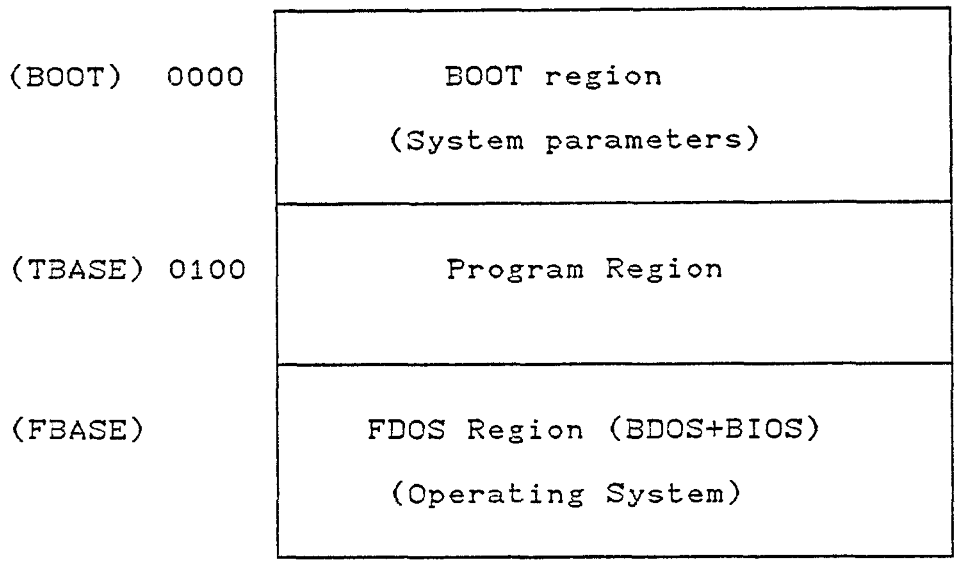
Figure 1

Sample DDL execution

```
STEP          PROCEDURE/RESULTS

1.  you       DDL

2.  Computer  MDBS.DDL VER X.X

              (C) COPYRIGHT 1979,1980, Micro Data Base Systems,

              Incorporated

              Reg # XXXXX

              your name and address

3.  you       R

4.  Computer  FILENAME

5.  you       SAMPLDDL.DDL

6.  you       L

7.  Computer  listing of sample data description

8.  you       BYE

9.  Computer  A>

(return to Monitor)
```

II. WALK-THROUGH OF PHASES 0-4 AND A-D

Memory Layout Under CP/M (all addresses in hex)



All standard CP/M versions use : BOOT = 0000 and TBASE = 0100.

All examples in this documentation assume these standard BOOT and TBASE addresses. If you have a non-standard CP/M version, a specially assembled version of MDBS software must be obtained and the examples adjusted accordingly. To find FBASE, either find the value in your CP/M manual or use the BASIC PEEK command for bytes 6 and 7 (and convert to hex). For example, with MBASIC:

```
MBASIC              enter BASIC
BASIC logon message
PRINT PEEK(&H6)
6                   decimal value of BOOT + 6
```

230 - E6
[illegible]

PRINT PEEK(&H7)

224 230 decimal value of BOOT + 7

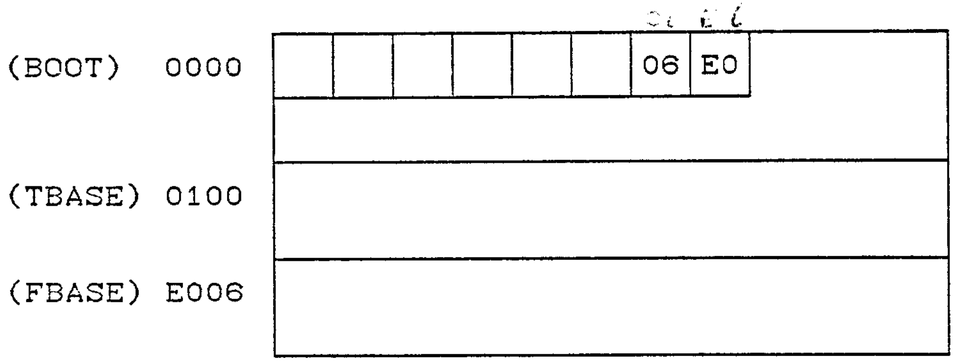
SYSTEM return to operating system

The contents of BOOT + 7 and BOOT + 6 give the value of FBASE. The values returned by the PEEKs must be converted to hex:

6/16 = 0 with remainder of 6 → 06: the hex contents of BOOT + 6

224/16 = 14 with remainder of 0 → E0: the hex contents of BOOT + 7

So FBASE in this example is E006.



Phase 0

. You will need to refer to the list of files on each diskette during the phases that follow. These phases make use of various files furnished on the MDBS diskette(s). To use a file of a diskette that is not on the currently active drive, you will need to indicate on which drive the file resides (e.g., in Step 1 of Phase 1: if the file RLC.COM is on drive D, but drive A is active, then you type D:RLC).

Phase 1

Step	Example	Comments
1.	>RLC	Load and execute RLC
2.	INPUT	
3.	DDL.IHF	Use the relocatable MDBS.DDL as input to the RLC relocator
4.	LOAD ADDRESS	
5.	0100,1F00	RLC produces an executable form of MDBS.DDL in memory starting at location 0100+1F00 = 2000. This executable form is ORGed at the first address given: 0100. It is necessary to produce this executable form at 2000 (rather than 0100) to avoid overwriting RLC which is executing at 0100. (The size of RLC itself can be determined by using RLC with file RLC.IHF as input).
6.	3611 [illegible]	The value returned (it may differ from the one shown here) is the high memory address of the executable form of MDBS.DDL.
7.	>DDT	In order to save the executable form of MDBS.DDL it must be moved to begin at 0100, so execute the CP/M routine: DDT.

8.	-M2000,5511,0100	Move the memory contents of 2000 through the high memory address (2000+3611-0100 = 5511) downward to be positioned at 0100.
9.	-control-C	Hit control-C to return to the operating system.
10.	>SAVE 56 DDL.COM	Save the executable form of MDBS.DDL in a 56 sector file called DDL.COM ((5511-2000)/0100 → ~56 sectors).

Phase 2:

Relocate MDBS.DMS at 7400 for MBASIC (BASIC version 5.xx) and relocate MDBS.DMS at 6800 for OBASIC (BASIC version 4.xx). An example of the former is shown here. These are suggested addresses (for BASIC 5.1 the suggested address is 7600); other addresses are also workable.

Step	Example	Comments
1.	>RLC	Load and execute the relocator.
2.	INPUT	
3.	DMS.IHF	Use the relocatable MDBS.DMS as input to the relocator.
4.	LOAD ADDRESS	
5.	7400	RLC produces an executable form of MDBS.DMS in memory starting at location 7400 and ORGed at 7400.
6.	BFD8 ~~CAE2~~ CCEL	High memory address of the executable form of MDBS.DMS (may

differ from the value shown here).

Phase 3: Not Needed

Phase 4:

Step	Example	Comments
1.	>MBASIC /M:&H73FF 77FF	The hex address following the H is usually 1 less than the DMS.COM ORG (e.g., 7400-1 = 73FF). If OBASIC (BASIC ver. 4.xx) is used the hex address is usually 101 less than the DMS.COM ORG (e.g., 6800-101 = 66FF). In any event, this hex address is the highest memory point for BASIC.
2.	BASIC logon message	
3.	POKE &H7406,&H05 ✓	
	POKE &H7407,&HE0 E6 7806 7807	Set the high address for the DMS buffers to E005, where E005 = E006-1. In general one usually changes the contents of 7407 and 7406 (i.e., the 6th and 7th byte of MDBS.DMS - the high memory address) to be FBASE-1 (Recall the earlier illustration for finding FBASE). If your ORG from step 5 of Phase 2 is, say, 7600 then you would POKE the high buffer address at 7606 and

7607, etc.

4. SYSTEM — Return to the monitor.

5. >SAVE 192 BASDMS.COM — Save the BASIC interpreter with attached executable form of MDBS.DMS in the 192 sector file called BASDMS.COM ((BFD8-0100)/0100 → ~192 sectors. Note that you can compute this while in BASIC by typing PRINT(&HBFD8&H100)/&H100)).

[illegible]AE2

[illegible]

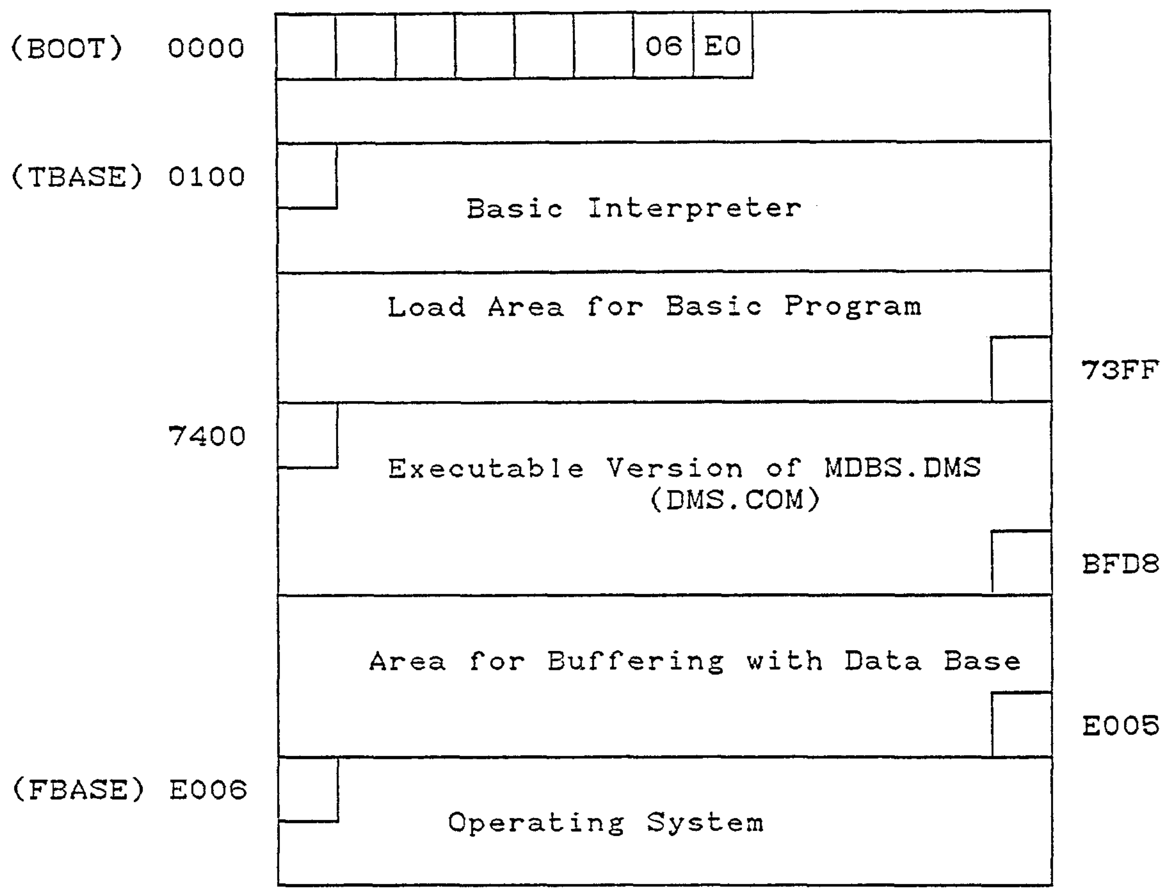
C4E2

CCF2

[illegible]

CBC2

Memory Contents at end of Phase 4:


(BOOT) 0000
06
E0
(TBASE) 0100
Basic Interpreter
Load Area for Basic Program
73FF
7400
Executable Version of MDBS.DMS (DMS.COM)
BFD8
Area for Buffering with Data Base
E005
(FBASE) E006
Operating System

Notice that the selection of an ORG for DMS.COM controls the tradeoff between the area available for a BASIC program vs. the area available for data base buffering. Setting the ORG lower than 7400 provides less space for a BASIC program but more space for buffering (and therefore faster processing). Setting the ORG higher than 7400 permits longer BASIC programs, but less space for buffering.

For instance, the sample program provided with HDBS is too long to be loaded if HDBS is ORGed to 7400. To accomodate this very long program, HDBS should be ORGed to a higher address (e.g., 7800 for MBASIC, 7200 for OBASIC).

The Book-Keyword Application[1]

Phase A:

Step	Example	Comments
1.	SAVE 1 SAMPLEDB.DB	Create the file SAMPLEDB.DB on the first drive (e.g., drive A). Note that SAMPLEDB.DB is the name specified in the SAMPLDDL.DDL DDL source file. (In HDBS, MEDDB.DB is used instead of SAMPLEDB.DB.)

Phase B:

Step	Example	Comments
1.	>DDL	Suppose the executable form of MDBS.DDL is on a file called DDL.COM. Execute it.
2.	MDBS header	
3.	R	Instruct DDL to read a file.
4.	FILENAME	
5.	SAMPLDDL.DDL	Read the SAMPLDDL.DDL file which contains the DDL source for the book-keyword application.
6.	xxxBYTES	Size of file just read.
7.	DDL	Run the analyzer.

[1]For HDBS this is a Doctor-Patient application.

.
.

8.	DDL PROCESSING COMPLETE	No errors in the DDL specification.
9.	BYE	Return to the system.

Phase C:

An application program has already been written and is provided in the SAMPLE file.

Phase D:

Step	Example	Comments
1.	>BASDMS	Execute the extended BASIC interpreter saved in Phase 4.
2.	LOAD "SAMPLE.BAS"	Load the application program. If there is insufficient memory to load, the DMS must be ORGed to a higher address (see Phase 2).
3.	Edit SAMPLE.BAS so that	the value asigned to AO at the start of the program is the ORG of the executable form of MDBS.DMS (Step 5, Phase 2). If you are using MBASIC and ORG = 7400, then you should have AO = &H7400. (If you are using version 4.0 BASIC, then USR8 should be set to the DMS ORG address.) See the more detailed notes that follow in section VI.

4. <u>RUN</u> Run the application progam.

III. GENERATING A NEW RELOCATOR

To produce a relocator at an address other than 0100H follow the sequence of steps shown in Figure 3.

Figure 3

Producing an Executable Relocator

Step		Procedure	Example
1.	you	RLC	RLC
2.	Computer	INPUT	
3.	you	RLC.IHF	RLC.IHF
4.	Computer	LOAD ADDRESS	
5.	you	yyyy (,zzzz)	6000
6.	Computer	tttt	
(Return to Monitor)		>	

Explanation:

This sequence of steps produces an executable form of RELOC.REL <u>in memory</u> starting at memory location yyyyH + zzzzH and ORGed at yyyyH (the "zzzzH" is optional). Executing RLC.COM results in its loading and subsequent execution at 0100H and the user must be careful not to have a memory conflict between RLC.COM and the newly formed RLC which will be placed at yyyyH + zzzzH.

The user may then want to save this program.

IV. MDBS.DDL PATCHING

MDBS.DDL consists of a program region and work area region. These are contiguous and the work area immediately follows the program area. The size of the work area can be increased or decreased through a patch to the system.

There are several addresses that the user should be aware of in MDBS.DDL. Figure 4 maps out these areas. A brief description of each item follows. All address patches should be made with normal 8080 conventions, i.e. the lower 8 bits of the address precede the higher 8 bits.

(a) Initial Entry Point

Upon entry at this point, all program variables and regions are either physically or logically re-initialized. Registers are not saved but the entering stack is preserved.

(b) BDOS Jump (Default 0005 hex)

This is the address of the disk management and peripheral processing routines.

(c) CP/M Warm Entry Jump (Default 0000 hex)

This address is the warm entry point to CP/M.

(d) FCB Location (Default 005C hex)

This points to the File Control Block area.

(e) BUFF Location (Default 0080 hex)

This points to a 128 byte buffer area.

(f) Reserved Regions

This area is reserved for internal use by the MDBS.DDL routines. It should not be altered.

(g) Echo Toggle (Default 01 hex)

This byte is checked to see if the user wants to have MDBS.DDL echo input to the output device. If the byte value is zero, echoing will take place. If it is the value one, no echoing will be performed.

(h) Size of BASIC Integer (Default 02 hex)

The value in this field gives the number of bytes required to store a numeric element in Microsoft BASIC. This value should be 2.

(i) Size of BASIC Single Precision Variable (Default 04 hex)

This field must contain the size of a Microsoft BASIC single precision variable. This value is 4.

(j) Last Word of Memory (Default 0000 hex)

The address stored here gives the last available word of memory that the MDBS.DDL program may use. Note that MDBS.DDL uses all memory starting from its load address up to the value in this field. Needless to say, the user should make sure that the last word of memory is physically beyond the end of the program. If this field is zero, a default of TBASE - 1 is assumed.

(k) Screen Control Byte (Default OB hex)

This byte should have one of the following values:

Screen Width	Byte Value
less than or equal to 64 characters	11 (OB hex)
greater than or equal to 80 characters	15 (OF hex)
64 to 80 characters per line (call it N)	greatest integer less than or equal to: N/5 -1

(l) Re-entry Point

The user may re-enter MDBS.DDL while preserving all program variables and regions by issuing a jump to this address.

Figure 4

MDBS.DDL ADDRESSES

Address	Description	Default Value
0100 (hex)	Initial Entry Point	--
0104 (hex)	BDOS Jump	0005 (hex)
0107 (hex)	CP/M Warm Entry	0000 (hex)
0109 (hex)	FCB Location	005C (hex)
010B (hex)	BUFF Location	0080 (hex)
010D-0126	Reserved	--
0127 (hex)	Echo Toggle	01 (hex)
0128 (hex)	BASIC Integer Size	02 (hex)
0129 (hex)	BASIC Single Precision Size	04 (hex)
012A (hex)	Last Word of Memory	0000 (hex)
012C (hex)	Screen Control Byte	11 (hex)
0139 (hex)	Re-Entry Point	--

V. MDBS.DMS PATCHING

MDBS.DMS consists of a program region, table region, page region and defined block region all of which are normally stored contiguously in memory. The other regions are application dependent. These sizes can be computed, approximately, by the formulas given in Section III.C of the MDBS.DMS User's manual.

Figure 5 maps out addresses of interest to the user of MDBS.DMS. The user may alter these. A brief description of each follows. (Refer to your CP/M and MICROSOFT BASIC-80 manuals for details.)

(a) Data Management System Entry Point

The user enters at this point to execute all of the DMS commands except for DEFINE and EXTEND.

(b) DEFINE and EXTEND Entry Point

To define a data block, the user issues calls to DEFINE and EXTEND which are entered at this location.

(c) Last Word of Memory (Default 0000 hex)

The address stored here gives the last available word of memory that MDBS.DMS may use for storing tables and page buffers. If this value is zero, a default of TBASE-1 is used.

(d) First Word of Memory (Default 0000 hex)

The address stored here gives the first word of memory available for the MDBS tables and page buffers. If a value of 0000 is present, the first word after the last word of the DMS routines will be used.

(e) CP/M Warm Entry Jump (Default 0000 hex)

This address is the warm entry point to CP/M.

(f) BDOS Jump (Default 0005 hex)

This is the address of the disk management and peripheral processing routines.

(g) FCB Location (Default 005C hex)

This points to the File Control Block area.

(h) BUFF Location (Default 0080 hex)

This points to a 128 byte buffer area.

(i) Size of BASIC Integer (Default 02 hex)

The value in this field gives the number of bytes required to store a numeric element in Microsoft BASIC. This value should be 2.

(j) Size of BASIC Single Precision Variable (Default 04 hex)

This field must contain the size of a Microsoft BASIC single precision variable. This value is 4.

(k) Reserved Region

This area is reserved for internal use by the MDBS.DMS routines. It should not be altered.

Figure 5

MDBS.DMS ADDRESSES

Load Address +	Description	Default Value
0000 (hex)	DMS Entry Point	--
0003 (hex)	DEFINE and EXTEND Entry Point	--
0006 (hex)	Last Word of Memory	0000 (hex)
0008 (hex)	First Word of Memory	0000 (hex)
000B (hex)	BDOS Jump	0005 (hex)
000E (hex)	CP/M Warm Entry Jump	0000 (hex)
0012 (hex)	BUFF Location	0080 (hex)
0014 (hex)	BASIC Integer Size	02 (hex)
0015 (hex)	BASIC Single Precision Size	04 (hex)
0016-002C	Reserved	--

VI. INTERFACING MICROSOFT BASIC-80

MDBS.DMS is callable from MICROSOFT BASIC. Below we show how such calls are made from the various versions of MICROSOFT BASIC.

a. USR Calls (for use with Version 4.xx BASIC)

To call the MDBS.DMS system from MICROSOFT BASIC using USR calls, the following procedure must be used. First the user must define the appropriate routine addresses. For example:

```
DEF USR8 = DMS Org Address
DEF USR9 = DMS Org Address + 3
```

Here USR8 will be the general DMS call and USR9 will be the call used with the DMS routines DEFINE and EXTEND.

To use USR8, a typical call would look like:

```
C$ = "FFM, SET3"   : E0% = USR8(C$)
```

A status value is returned in the <u>integer</u> variable E0%.

Calls to DEFINE and EXTEND to set up data blocks are a bit more complicated. Here the user must set up a block of memory with addresses pointing to:

1. The routine and block name string
2. An integer variable giving the number of arguments
3. The address of argument one
4. The address of argument two

   etc.

For example, to set up a data block named OPEN for variables F$, N$, P$ and R$ we might use the following scheme. Due to a requirement of the VARPTR function, the variables F$, N$, P$ and R$ must be initialized before this code is executed.

```
100  B% = &H6700[1]
101  N% = 4
102  C$ = "DEFINE,OPEN"
103  I% = VARPTR (C$)
104  GOSUB 200
105  I% = VARPTR (N%)
106  GOSUB 200
107  I% = VARPTR (F$)
108  GOSUB 200
109  I% = VARPTR (N$)
110  GOSUB 200
111  I% = VARPTR (P$)
112  GOSUB 200
113  I% = VARPTR (R$)
114  GOSUB 200

200  I1% = (I% AND 32512) / 256
210  IF I% < 0 THEN I1% = I1% + 128
220  I2% = I% AND 255
230  POKE B%, I2%
240  B% = B% + 1
250  POKE B%, I1%
260  B% = B% + 1
270  RETURN
```

The actual call to DEFINE would follow the above and would look like:

```
E0% = USR9 (&H6700)
```

b. CALL Statement Calls (for use with Version 5.xx BASIC)

A call to Find First Member (FFM) would look like:

```
C$ = "FFM, SET3"  : CALL A0(E0%, C$)
```

where A0 holds the DMS ORG address. A status value is returned in an integer variable E0%.

[1]The variable B% must be set to a memory address which is not used by either the BASIC program, the operating system or the DMS routines. If the high address of BASIC is set to 66FF (Step 11,Phase 4) , then a good choice for B% is 6700 (providing MDBS.DMS is ORGed to the suggested 6800 in Phase II). See Section VII,7.

The user can define data blocks which are a collection of BASIC variables that can be accessed by the DMS routines. To define a data block, a call to routine DEFINE is made like:

```
          N% = 4
          C$ = "DEFINE, BLOCK"  : CALL A1(E0%,C$, N%, F$, N$, P$, R$)
```

Here A1 is ORG+3 (as explained in Section III of this manual). This call will result in a data block named BLOCK having variables F$, N$, P$, and R$. For the DEFINE and EXTEND routines, the third parameter (here N%) specifies how many variables follow N%. This variable must be an integer variable.

VII. MISCELLANEOUS

1. Replicated items (see Section II.D.5 of the User's Manual) are given and returned to MDBS.DMS from BASIC using arrays. Since MICROSOFT BASIC may relocate arrays during program execution and MDBS.DMS assumes that the addresses supplied in the DEFINE and EXTEND calls stay in effect throughtout the run, extreme caution should be employed when using arrays with MDBS.DMS. Refer to the VARPTR discussion in the MICROSOFT BASIC manual for details.

2. Logical, Binary and Integer Item Types are supplied from (and to) BASIC through INTEGER variables. REAL Item Types are supplied through Single Precision and Double Precision variables. In the DDL Analyzer, a blank item size for a REAL variable results in a single precision default. Double Precision variables result when the appropriate item size (8) is specified.

3. When retrieving character strings from a data base, the user should make sure that the destination string variable has been previously initialized to a length sufficiently large to hold the incoming string. If this is not done, the incoming string will be truncated. Further, the string should not be initialized by setting it equal to a program literal. This can cause the literal in the program to be modified with undesirable results. It is recommended that the string variable be initialized with the SPACE$ function. Alternatively, the FIELD command may be used to insure that adequate space has been allocated for a string.

4. Note that two byte integer variables run from -32767 to 32767 and not from -32768 to 32767.

5. In the DDL Analyzer, a new data base can be initialized on any drive by specifying an appropriately qualified file name on the FILES specification.

   In the DMS OPEN command a drive may be included in the data base file name specification. The drive specified is assumed to be the starting drive for the data base (corresponding to Drive 1 in the DDL DRIVE specifications).

6. In both the DDL Analyzer and the DMS, when no drive is specified for a file, the logon drive is the default.

7. The DMS package comes in two forms, one that accepts USR calls (for use with BASIC Versions 4.0 and 5.0) and one that accepts the CALL calls (for use with BASIC version 5.0). To obtain both copies an extra handling and preparation charge will be required. The version 4.0 user must remember to allocate room for the POKE values used in calls to DEFINE and EXTEND. A good place is immediately after BASIC but before the MDBS.DMS package. For example, in Phase 4 Step 11 the end of BASIC was specified as 73FF (hex). Version 4.0 users would specify a lower value (e.g.,66FF) to allocate the necessary space.

8. DEFINE and EXTEND calls are limited to 4 variables per call. Successive calls to EXTEND should be used to set up long data blocks.

9. All arrays should be declared at the beginning of a BASIC program (prior to any calls to the data base management system).

10. Additional main memory can be freed by killing the BASIC file buffers (if they are not needed by your BASIC program). This is accomplished by appending Step 1 of Phase 4 as follows:
>MBASIC /M:&H73FF /F:0.

11. Filenames in the DDL Analyzer are defaulted to NAME.TXT.