SYS.PLI

MDBS - Digital Research PL/I Interface Notes

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

September 1980

OUTLINE

MDBS - PL/I Interface Notes

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 Digital Research PL/I or related products.

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. RLC.COM - An absolute relocator
4. RLC.IHF - A relocatable (hex format) relocator
5. DDL.IHF - A relocatable (hex format) version of the MDBS.DDL system
6. SAMPLDDL.DDL - Sample DDL file for use with SAMPLE.PLI
7. DMS.REL - A relocatable version of the MDBS.DMS system
8. SAMPLE.PLI - Sample PL/I program for use with SAMPLDDL.DDL

MDBS.DDL is your personalized Data Definition Language Analyzer/Editor and MDBS.DMS your Data Management System. DDL loads at 100H and extends to approximately 3400H. It uses standard CP/M I/O entry points. If you need to change the DDL load address, Sections II and III outline use of the relocator supplied with the MDBS system. If you are using nonstandard I/O entries or require other patching, refer to Section IV for patching procedures.

MDBS.DMS is supplied in a relocatable format compatible with the object files produced by the PL/I compiler. The program area for the MDBS.DMS system is approximately 4400 (hex) bytes long. Additionally, memory is required for the various tables and buffers used by the system. This memory space is allocated in the calling PL/I program to maximize the amount of memory available to the MDBS.DMS system subject to the constraints of the operating environment. Increased memory passed to the MDBS system will produce greater system throughput.

In Section VI we show how to call the DMS routines from Digital Research PL/I. Section VI describes the procedure to link the MDBS.DMS system with a compiled PL/I program.

Finally, in Section VII we provide some general notes for the PL/I user of the MDBS package.

You can run your version of DDL by following the procedure given in Figure 1. At the end of Step 2 you are in the DDL program. Refer to Section III.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 INVNTRY.DDL. Step 6 results in the listing of the sample description. Step 8 returns control to the operating system.

Figure 1

Sample DDL execution

STEP		PROCEDURE/RESULTS
1.	you	DDL
2.	Computer	MDBS.DDL VER X.X
		(C) COPYRIGHT 1979, Micro Data Base Systems, Incorporated
		Reg # XXXXX
		your name and address
3.	you	R
4.	Computer	FILENAME
5.	you	INVNTRY.DDL
6.	you	L
7.	Computer	listing of sample data description
8.	you	BYE
9.	Computer	A>

(return to Monitor)

II. RELOCATING MDBS.DDL

To relocate and produce an executable form of MDBS.DDL we have provided an executable relocator and a relocatable form of the relocator. RLC.COM loads at 0100H. To use it merely enter the sequence of lines shown in Figure 2 at your terminal. After performing this sequence of steps you may want to save the file. If a relocator is needed at an address other than 0100H refer to Section IV.

Figure 2

Producing an Executable Form of MBDS.DDL

Step		Procedure	Example
1.	you	RLC	RLC
2.	Computer	INPUT	
3.	you	MDBS.DDL	MDBS.DDL
4.	Computer	LOAD ADDRESS	
5.	you	yyyy (,zzzz)	2D00
6.	Computer	tttt	
(return to Monitor)		>	

Explanation:

This sequence of steps produces an executable form of MDBS.DDL in memory starting at memory location yyyyH + zzzzH and ORGed at yyyyH. The ",zzzz" is The high memory address of the routine is indicated by "tttt". The user should insure that the relocator and the program locations are not in conflict with one another.

Following this procedure, the user may want to make patches to the executable program (as outlined in Section II.C.3 of the MDBS user's manual and in Sections IV and V of this manual) and then save the resulting executable program.

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 RLC.IHF in memory 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) Last Word of Memory (Default 0BFFF 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.

(i) Screen Control Byte (Default 0B hex)

This byte should have one of the following values:

Screen Width	Byte Value
less than or equal to 64 characters	11 (0B hex)
greater than or equal to 80	15 (0F hex)

characters	
64 to 80 characters per line (call it N)	greatest integer less than or equal to: N/5 -1

(j) 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)
012A (hex)	Last Word of Memory	BFFF (hex)
012C (hex)	Screen Control Byte	11 (hex)
0139 (hex)	Re-Entry Point	--

V. INTERFACING PL/I

A. Data Types

The following correspondence exists between Digital Research PL/I data types and the data types of MDBS.DMS:

PL/I	MDBS	Comments
Fixed Binary 1-7	BIN 1	
Fixed Binary 8-15	BIN 2	INT may also be used
Bit 1-8	BIN 1	
Bit 9-16	BIN 2	
Char(n)	CHAR n	Note: if VARYING is used increment n by one.
Fixed Decimal n	IDEC n	
Floating Point Binary	REAL	
Pointer,Entry,Label	not used	

The MDBS LOGical data type has no correspondence in PL/I and is not supported under the PL/I version of the MDBS.DMS system. An additional data type (Internal Decimal, or IDEC) has been defined to represent fixed decimal fields. The item size "n" on an IDEC entry cooresponds to the precision "p" of the item, which may vary from 1 to 15.

B. Command Strings

The MDBS User's Manual shows calls to the various DMS functions in a format similar to that used with North Star BASIC. The conventions used for PL/I are similar, except that PL/I permits definition of structures which correspond to record types in the data base. This obviates the need for data blocks, as described in the following section. Thus, the command string need not have a data block name specified in such commands as CRS, FMSK, etc.

C. Data Communication

Due to the non-record oriented data structures of BASIC and other languages, it was necessary to include "data block" capabilities in MDBS to facilitate communication between the host language and the MDBS system. Since PL/I has record oriented data structures, use of the data block facilities are not needed. Instead, an additional parameter is used in calls to MDBS.DMS to pass a pointer to the record area to be used for transfer of data. This results in a more natural data representation scheme in the PL/I program and eliminates some overhead from the MDBS.DMS system. Thus, a call to the CRS routine might look like:

```
      err=DMSD('CRS,RECORD',struct);
```

where err is a Fixed Binary (15) variable, the first argument is of type CHAR VARYING and struct is a pointer to a structure which corresponds to the data record "RECORD" in the data base. DMS procedures that do not expect data can be invoked via a call to entry point DMS of the form:

```
      err=DMS('FNM,SET');
```

E. The SETPBF Routine

The SETPBF entry point into the MDBS.DMS system allows the programmer to easily allocate memory for use by the MDBS.DMS system. SETPBF must be called before any other DMS command. Memory is allocated for use by MDBS.DMS by allocating storage in the calling program's dynamic storage area. This area is used by MDBS.DMS to store tables and buffers and MUST NOT BE ALTERED by the user's program. Since the MDBS.DMS system uses a dynamic buffering scheme, it is important that as much memory as possible be allocated for use by the MDBS system (we recommend that at least 4000 bytes be allocated). Since the amount of memory available to the MDBS.DMS system can radically affect the throughput of the system (more memory implies more disk buffers which implies fewer disk accesses), SETPBF has been implemented to make it easy for a programmer to allocate all of the dynamic memory for use. The declaration and calls for SETPBF could be:

```
dcl SETPBF entry (ptr,fixed(15));
dcl MAXWDS returns(fixed(15));
dcl ALLWDS entry(fixed(15)) returns (ptr);
dcl WDS fixed(15);
 .
 .
 .
WDS=MAXWDS()/2;  */Allocate half of available space for DMS/*
CALL SETPBF(ALLWDS(WDS),WDS*2); */Second parameter is a byte count/*
```

F. The OPEN Command

The MDBS User's Manual indicates that the OPEN command requires a data block containing four parameters - a data base file name, a user name, a password and a read/write status. For use with PL/I, the following record must be defined in the declaration of the the PL/I program:

```
dcl 1 openblk,
      2 dbname char(14)
      2 user char(16)
      2 password char(12)
      2 rwstat char(4)
```

The length of the data items must be exactly as shown; the values are of course application dependent. In most applications, the values for the user name and password are input with GET statements. With an OPEN block such as we have described, the call to open the data base would be:

```
err=DMS('OPEN',openblk)
```

VI. LINKING THE PL/I PROGRAM AND MDBS.DMS

The MDBS.DMS system may be linked with your compiled source program by using the LINK-80 linker. All that is necessary is to load the DMS and your program as shown below ("PGM" is the name of the file containing your compiled program). Note that any of the standard Digital Research LINK-80 options may be used. The process of linking the MDBS.DMS system may take several minutes due to the size of the DMS and the use of the LINK-80 [A] load option.

```
LINK PGM,DMS[A]
```

VII. GENERAL NOTES

1. Replicated items (see Section II.D.5 of the User's Manual) are given and returned to MDBS.DMS from PL/I using arrays.

2. The DEFINE and EXTEND commands are not needed in the PL/I version of the MDBS.DMS system since PL/I is oriented toward the definition and use of record data structures (which correspond to data base records).

3. 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).

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

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