



# **MIMER**

## **User's Guide for VMS**

**Version 7.3**

**Copyright © 1996 Sysdeco Mimer AB**

MIMER version 7.3 User's Guide for VMS

December, 1996

Copyright © 1996 Sysdeco Mimer AB.

Published by Sysdeco Mimer AB,

P.O.Box 1713,

S-751 47 Uppsala, Sweden.

Tel +46(0)18-18 50 00.

Fax +46(0)18-18 51 00.

Internet: <http://www.mimer.se>

Produced by Sysdeco Mimer AB, Uppsala, Sweden.

All rights reserved under international copyright conventions.

The contents of this manual may be printed in limited quantities for use at a MIMER installation site. No parts of the manual may be reproduced for sale to a third party.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	
1.1	Acronyms and trademarks .....	1-1
<b>2</b>	<b>GENERAL INFORMATION</b>	
2.1	Structure of a MIMER installation .....	2-1
2.1.1	The [MIMER7] runtime directory .....	2-1
2.1.2	Logical names .....	2-2
2.2	Multiple databases .....	2-3
2.3	Single versus multi-user mode .....	2-3
2.3.1	General considerations.....	2-3
2.3.2	Running in single-user mode .....	2-4
2.3.3	Multiple multi-user systems .....	2-4
2.4	Using MIMER in client/server mode .....	2-5
2.5	Starting MIMER modules.....	2-6
2.5.1	Finding the executable files.....	2-6
2.5.2	Logging in to MIMER modules .....	2-6
2.5.3	The MIMINI.DAT file.....	2-6
2.6	Databank organization.....	2-7
2.6.1	General features .....	2-7
2.6.2	Extending databanks .....	2-8
2.6.3	Single-user systems .....	2-8
2.6.4	Multi-user systems .....	2-9
2.6.5	Conversion between single and multi-user systems.....	2-9
2.6.6	Copying and moving databank files.....	2-10
2.6.7	Dropping databanks or shadows .....	2-10
2.7	Error codes .....	2-11
2.7.1	MRS errors .....	2-11
2.7.2	MIMER/DB system access errors .....	2-12
2.7.3	Direct access I/O errors.....	2-13
2.7.4	Sequential access I/O errors .....	2-14
2.8	Developing applications .....	2-15
2.8.1	Using the old call level interface.....	2-15
2.8.1.1	Writing applications in C.....	2-15
2.8.1.2	String parameters in FORTRAN.....	2-15
2.8.2	Optimizing compilers.....	2-16
2.8.3	Linking applications .....	2-17
2.9	Using alternative MIMER installations.....	2-18
2.10	Converting a version 7.1 or version 7.2 database.....	2-18
2.11	Converting a version 5 database .....	2-18
2.11.1	Upgrading databanks .....	2-19
2.11.2	Using version 5 applications with MIMER version 7 .....	2-19
<b>3</b>	<b>MODULE-SPECIFIC INFORMATION</b>	
3.1	MIMER/CONV .....	3-1
3.1.1	Accessible files.....	3-1

3.1.2	Running the conversion .....	3-1
3.1.3	File type conventions .....	3-2
3.2	MIMER/DB .....	3-3
3.2.1	Accessible files in MIMER/DB .....	3-3
3.2.2	The MIMER/DB runtime libraries .....	3-4
3.3	MIMER/ESQL .....	3-5
3.3.1	Accessible files.....	3-5
3.3.2	VAX/VMS compilers supported .....	3-5
3.3.3	ESQL command syntax.....	3-6
3.3.4	File handling conventions.....	3-6
3.3.5	Type definitions .....	3-6
3.3.6	Building the simple EXAMPLE program.....	3-7
3.3.7	Building the DSQSAMP example program.....	3-7
3.4	MIMER/FMD .....	3-8
3.4.1	Accessible files.....	3-8
3.4.2	File handling conventions.....	3-8
3.4.3	Function keys in the FM editor.....	3-8
3.5	MIMER/FMR.....	3-10
3.5.1	Accessible files.....	3-10
3.5.2	Supported terminal types.....	3-10
3.5.3	Function keys in MIMER/FMR .....	3-10
3.5.4	Set keypad mode .....	3-12
3.5.5	Linking applications to MIMER/FMR.....	3-13
3.6	MIMER/ISQL .....	3-14
3.6.1	Accessible files in MIMER/ISQL.....	3-14
3.6.2	Executing DCL commands.....	3-14
3.6.3	Keyboard interrupt .....	3-14
3.6.4	Function keys in the ISQL full screen editor.....	3-15
3.6.5	The MIMINI initiation file and ISQL/BSQL .....	3-16
3.7	MIMER/PGD .....	3-17
3.7.1	Accessible files.....	3-17
3.7.2	File handling conventions.....	3-17
3.7.3	Executing DCL commands.....	3-18
3.7.4	Keyboard interrupt .....	3-18
3.7.5	The MIMINI initiation file and PG.....	3-18
3.7.6	Code generators .....	3-18
3.8	MIMER/PGR .....	3-19
3.8.1	Accessible files.....	3-19
3.8.2	Linking a PG application.....	3-19
3.8.3	The PGRUN command procedure .....	3-19
3.9	MIMER/PI .....	3-21
3.9.1	Accessible files.....	3-21
3.9.2	Linking an application that uses MIMER/PI.....	3-21
3.9.3	Exit routines .....	3-21
3.10	MIMER/QF.....	3-22
3.10.1	Accessible files.....	3-22
3.10.2	Function keys in MIMER/QF .....	3-22
3.11	MIMER/QL.....	3-24
3.11.1	Accessible files.....	3-24
3.11.2	Executing DCL commands.....	3-24
3.11.3	Keyboard interrupt .....	3-24
3.11.4	The MIMINI initiation file and QL.....	3-25
3.12	MIMER/RG .....	3-26
3.12.1	Accessible files.....	3-26
3.12.2	Executing DCL commands.....	3-26
3.12.3	Function keys in MIMER/RG.....	3-27
3.12.4	Default file types.....	3-28
3.12.5	The MIMINI initiation file and RG .....	3-28

3.12.6	Linking RG applications.....	3-28
3.13	MIMER/SHD .....	3-29
3.13.1	Accessible files.....	3-29
3.13.2	Starting the editor.....	3-29
3.14	MIMER/SHR .....	3-30
3.14.1	Accessible files in MIMER/SHR.....	3-30
3.14.2	Initialization files .....	3-30
3.14.3	Terminal types.....	3-31
3.14.4	Selecting terminal type.....	3-31
3.14.5	Terminal definitions.....	3-31
3.14.6	Terminal-specific key sequences .....	3-31
3.14.7	The SH Screen Compiler .....	3-33
3.14.8	Linking application programs with SH.....	3-33
3.14.9	Runtime version check facility .....	3-34
3.15	MIMER/UTIL .....	3-35
3.15.1	Accessible files.....	3-35
3.15.2	File handling conventions.....	3-35
3.15.3	The MIMINI initialization file and MIMER/UTIL .....	3-35
<b>4</b>	<b>DATA TYPES USED IN MIMER</b>	
4.1	Internal DB representation .....	4-1
4.2	External data types supported by MIMER.....	4-2
<b>A</b>	<b>USING TAPE DEVICES AS SEQUENTIAL FILES</b>	
A.1	Writing data to magnetic tapes .....	A-1
A.1.1	Preparations .....	A-1
A.1.2	Writing to the tape unit.....	A-1
A.2	Reading sequential data from a tape.....	A-2
<b>B</b>	<b>EXIT ROUTINES</b>	
B.1	General considerations .....	B-1
B.2	Using standard error handling.....	B-2
B.3	Advantages of standard error handling.....	B-2



# 1 INTRODUCTION

This guide documents MIMER running on VMS and discusses machine-specific features of MIMER relevant to the users of the system. Machine- and site-specific features of the installation procedures are described in the *MIMER Installation Guide for VMS*. The machine-specific documentation does not describe how to use MIMER, and should be used as a supplement to the more generally applicable Reference Manuals for the separate MIMER modules.

This guide is intended for MIMER V7.3 on both VAX/VMS and Alpha/VMS.

**VAX**

Some parts of the text are marked with a VAX icon in the left margin to indicate that the text only applies to the VAX/VMS implementation of MIMER. This architecture dependent text is terminated by a solid rhomb. ◆

**AXP**

Similarly, an AXP icon in the left margin indicates that the text applies to the Alpha/VMS implementation of MIMER. ◆

---

**Note:** Several file and directory specifications are quite long in VMS, e.g. SYSS\$COMMON:[SYSLIB]MIMDB7M\_name.EXE. These may for aesthetic reasons be split over more than one line in this document. This does not imply that the names may contain spaces or be split over two lines in VMS.

---

## 1.1 Acronyms and trademarks

MRS	MIMER Release and Security.
API	Application Programming Interface.
ESQL	Embedded SQL
SMG	Screen Management Guidelines.
VMS	VMS is a registered trademark of Digital Equipment Corporation.

(All other trademarks are the property of their respective holders.)



## 2 GENERAL INFORMATION

### 2.1 Structure of a MIMER installation

#### 2.1.1 The [MIMER7] runtime directory

It is recommended that the person responsible for installing MIMER put the MIMER installation under a directory named MIMER7. However, the installer is free to give the directory any name. In the rest of this document, it is assumed that the root name of the MIMER runtime directory is MIMER7.

The MIMER7 directory contains four subdirectories of interest to MIMER users:

- [.EXEC] This subdirectory contains all executable (.EXE) files for all installed MIMER modules. The logical name MIMEXE7 points to this directory.
- [.LIBRARY] This subdirectory contains all libraries and data files that a MIMER user may want to read. The logical name MIMLIB7 points to this directory.
- [.VERSION] This directory contains one file for each module installed. The first line in each of the files specifies the installed version of the module. The files are updated automatically as new versions are installed.
- [.DOC] This directory contains on-line MIMER manuals in Portable Document Format (.PDF). Files in this format may be read using the Adobe Acrobat reader. The reader is available free of charge from Adobe (web site: <http://www.adobe.com>). There is currently no Acrobat reader available for VMS. Readers are available for Mac, PC and most UNIX platforms and the PDF files are portable so they may be copied to any platform for which Adobe provide a reader.

### 2.1.2 Logical names

To ensure proper operation of the MIMER system, the system-wide logical names listed below should have been defined. This is done by the command procedure found in the file MIMSETUP7.COM in the directory [MIMER7.LIBRARY]. The system administrator should define these logical names in the SYSTEM name table when the MIMER system is installed and when the machine is rebooted.

Logical name	Translation	Usage
MIMROOT7	disk:[MIMER7.]	This is a concealed logical name that gives the root level of the runtime tree. All other logical names that refer to the runtime tree use this logical name.
MIMDOC7	MIMROOT7:[DOC]	Contains on-line documentation.
MIMLIB7	MIMROOT7:[LIBRARY]	Contains all library (.OLB) -files and all data files to which users need R (read) -access
MIMEXE7	MIMROOT7:[EXEC]	Contains all executable (.EXE) files in the MIMER system. Users must have E (execute) -access to these files.
MIMKEY7	SYSS\$SPECIFIC:[SYSMGR] MIMKEY7.DAT	This file contains the MRS software key. The encoded data in this file allows or disallows usage of MIMER modules.
MIMERM7	MIMLIB7:MIMERM.OPT	This is an option file used when linking application programs to a MIMER/DB multi-user system (and MIMER/FM).
MIMERS7	MIMLIB7:MIMERS.OPT	This is an option file used when linking application programs to a MIMER/DB single-user system (and MIMER/FM).
MIMERMSH7	MIMLIB7:MIMERMSH.OPT	This is an option file used when linking application programs to a MIMER/DB multi-user system and MIMER/SH.
MIMERS77	MIMLIB7:MIMERS77.OPT	This is an option file used when linking application programs to a MIMER/DB single-user system and MIMER/SH.

Other logical names may be defined for convenience. Care must be exercised to ensure that any logical names defined for the user's own convenience do not conflict with other usages in the VMS system. A good policy is to start all MIMER-specific logical names with a prefix (such as 'MIM').

## 2.2 Multiple databases

A database is a collection of databanks, tables, MIMER users, etc. All of these objects are defined in the data dictionary, SYSDB. Each database has a unique name. The name of each database, and the location of its data dictionary file (SYSDB7.DBF) is entered in the MIMLIB7:SQLHOSTS.DAT file. No data may be shared between databases.

There are three mechanisms that determine which database an application will connect to:

- If a database name is given explicitly in the CONNECT SQL statement, this database will be used. Note that there is no way to give the database name explicitly if the application connects to the database using the level 2 interface (BEGIN2).
- If no explicit database name is given by the application, the logical name MIMER\_DATABASE will be checked. If this name is defined, the database that the logical name points to will be used.
- If none of the methods above finds a database name, the database defined in the DEFAULT clause in the MIMLIB7:SQLHOSTS.DAT file will be used.

The SQLHOSTS file can contain both remote and local databases. If a remote database is specified, the MIMER client/server interface will automatically be used (see Section 2.4).

## 2.3 Single versus multi-user mode

### 2.3.1 General considerations

The MIMER software is available in versions for single and multi-user installations. A single-user system permits only one user at a time to access a local database, although many users may be authorized to use the database (but not simultaneously). A multi-user system permits any number of users to simultaneously access the same database, up to limits set by the MRS license and parameters specified when the system is started.

Both single-user mode and multi-user mode applications may use the client/server interface. This interface is transparent to the application. It is possible for several single-user mode applications to use the same database, if that database is accessed with the client/server facility.

Databanks and application program source and object code are identical in both single and multi-user systems. Databanks for different systems are identified through the data dictionary (residing in the SYSDB databank) for the system in question. Application programs are linked with MIMER/DB shareable libraries for single or multi-user systems respectively. An application program end-user may not necessarily be aware of whether the program runs in single or multi-user mode.

### 2.3.2 Running in single-user mode

Most MIMER executable modules access MIMER/DB and are available in both single and multi-user versions. The last letter of the name of the executable file of these programs is either M or S. Thus the program ISQLM runs ISQL in multi-user mode, and UTILS is the single-user mode version of the UTIL program. During MIMER installation, the system manager decides whether to build the single and/or multi-user versions of the MIMER modules. The default is to build only the multi-user versions.

Since the MIMER/DB single-user and multi-user libraries are shareable images, a program that has been linked to the multi-user shareable library (MIMDB7M) can run in single-user mode, if the logical name MIMDB7M is redefined to point to MIMDB7S. For example, to run the UTILM program in single-user mode, the following commands can be used:

```
$ DEFINE MIMDB7M MIMDB7S
$ RUN MIMEXE7:UTILM          ! NOTE: UTILM, not UTILS
```

When the RUN command maps the shareable image MIMDB7M, it will read the MIMDB7S image instead. The MIMDB7M and MIMDB7S images are fully compatible. By using this method, the programs need only be linked once to be available in both single- and multi-user mode. Note, however, that as long as the MIMDB7M logical name points to MIMDB7S, all multi-user mode programs will run in single-user mode. To run multi-user mode applications, the logical name must be deassigned. For example, to run the UTILM program in multi-user mode again, use the following commands:

```
$ DEASSIGN MIMDB7M
$ RUN MIMEXE7:UTILM
```

To summarize, there are two methods of running a MIMER/DB application in single-user mode; using a specially linked .EXE-file, or by defining logical names. The advantage of the specially linked application method is that it is easy for any user to know whether single-user or multi-user mode is used. On the other hand, more program files must be created to support both single-user and multi-user mode.

The advantage of the logical name method is that only one version of each application is needed. However, if a user is alternating frequently between single-user and multi-user mode, defining the logical name for each invocation might be confusing.

Since most end-users will run multi-user mode only, the logical name method is recommended, since it avoids having to build all MIMER-modules in single-user mode. (The default is to install MIMER modules in multi-user mode only.)

### 2.3.3 Multiple multi-user systems

A multi-user system provides access for several users to *one* local database. If there are several local databases on a machine, each database will need a separate multi-user system. Each multi-user system can serve exactly one database at a time. The user need not be concerned with which multi-user system he is actually connected to, and need only supply a database name.

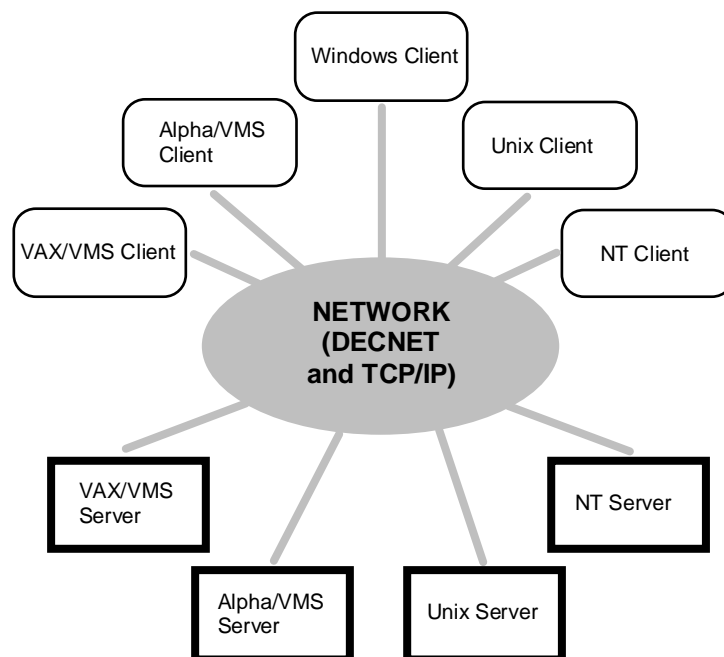
## 2.4 Using MIMER in client/server mode

The MIMER client/server interface is integrated into the MIMER/DB module. It is available to all installations that have the appropriate MRS licenses. All MIMER/DB applications may use the client/server interface without having to make any special provision in the application code.

The client/server interface is activated automatically whenever a remote database is specified. (A remote database is a database that is described under the REMOTE section in the SQLHOSTS file. See the *MIMER Installation Guide for VMS* for more information).

The MIMER client/server protocol is identical on all MIMER platforms. This means that a MIMER/DB client on any machine type may access MIMER/DB servers on all machine types where MIMER version 7 has been implemented.

An application may connect to several remote databases simultaneously. By using the SQL statement SET CONNECTION the application may switch between active connections. However, a transaction may only use one connection at a time.



## 2.5 Starting MIMER modules

### 2.5.1 Finding the executable files

All MIMER modules are started with the DCL command RUN, except for the ESQL module (see Chapter 3). All executable MIMER modules are found in the MIMEXE7: directory. Most modules usually exist in both single and multi-user mode (although the system administrator may have chosen to only build the multi-user versions to save disk space). The last character in executable files is either S or M, signifying single or multi-user modes, respectively. Some examples:

```
$ RUN MIMEXE7:BSQLM ! Runs the BSQL module in multi-user mode.  
$ RUN MIMEXE7:UTILS ! Runs the UTIL module in single-user mode.
```

---

**Note:** If the single-user version of a module has not been built (which is the default installation), it is still possible to run the multi-user version of the application in single-user mode. This is further described in Section 2.3.2.

---

The names of all executable files available within each module are listed in Chapter 3 under the relevant section.

### 2.5.2 Logging in to MIMER modules

MIMER/DB has a security system that requires all users to identify themselves before any database functionality becomes available. Most MIMER modules that use MIMER/DB have a login form, showing the MIMER logo and "Username" and "Password" prompts. In this login form, the user is given a maximum of three attempts to log in before the program terminates. The MIMER username and password provided by the MIMER system administrator are to be used to log in, not the VMS username/password.

If the system administrator has defined an OS\_USER, with the same name as the VMS user, it is not necessary to specify a username or password when logging in to the MIMER system. Just pressing the return key when the "Username" prompt is shown will log in the user as the corresponding OS\_USER. How an OS\_USER is defined is described in the *MIMER/SQL Reference Manual* (CREATE IDENT).

### 2.5.3 The MIMINI.DAT file

Most MIMER modules that show the login form previously mentioned will also accept login information from the MIMINI file. The default name for the file is MIMINI.DAT in the current default directory. The initialization file may also be referenced by the logical name MIMINI. In addition to the Username and Password details which may be stored in this file, some modules accept additional information (see the *MIMER/QL Reference Manual* for an example). This is described further in the section for each module in Chapter 3.

## 2.6 Databank organization

The following terms are important in the discussion that follows:

<b>database</b>	A database is a collection of tables, databanks, idents, domains, etc. All of these objects are defined in the data dictionary, SYSDB. A VMS system may have several databases operating simultaneously, <i>but no information is shared between them</i> . Each database has a unique name, registered together with the location of its SYSDB databank in the file MIMLIB7: SQLHOSTS.DAT.
<b>databank</b>	A databank corresponds to a VMS file. If the databank is shadowed, there will be one VMS file for each shadow. A database may contain several databanks.
<b>table</b>	Tables (or relations) hold all the information in the relational database. A table may not be split over several databanks, but a databank may hold several tables.
<b>shadow</b>	A MIMER databank may have one or several shadows. A shadow is a copy of the original (master) databank and is continuously updated by MIMER/DB. If the master databank is lost, it is possible to continue operations from the shadow databank without stopping the multi-user system. A databank must have the TRANS or LOG option to be shadowed.

If database shadowing is being used, the *MIMER Shadowing Reference Manual* should be consulted.

### 2.6.1 General features

Databank files are always accessed by direct access I/O routines in the database manager. File names for all databanks are stored in the data dictionary (SYSDB) in the exact form that they are entered when the databanks are created. Users are strongly advised to give full unambiguous file specifications or logical names when creating databanks. Use of logical names can simplify databank relocation.

If no explicit device or directory is specified when a databank is created, the file will be placed in the same directory as the SYSDB databank.

When creating and addressing databank files, the default file type '.DBF' will be used by the system if no type is specified.

## 2.6.2 Extending databanks

All MIMER databank files will be extended automatically when the file is too small to hold the desired information. The extension will fail if the disk is full, or if the owner of the databank file has no disk quota available. By default, the databank files will be extended by 100 VMS blocks at a time.

The extend size for a file can be altered by giving the DCL command:

```
$ SET FILE/EXTENSION=extensionsize file.DBF
```

This command should be given when no MIMER/DB system (single- or multi-user mode) is using the file.

Having a large file extension size will result in fewer extension operations on a file and will improve performance slightly if the databank is growing rapidly. However, it will waste disk space since the file will generally contain more unused disk blocks.

Having a small file extension size will save disk space since the file size follows the actual size more closely. However, this may cause disk fragmentation. In addition, if the databank is growing, file extension operations have to be done more frequently, thus decreasing performance.

The optimum performance will be achieved if the file size can be estimated when the databank is created. In this case, no extend operations will be required, and disk fragmentation will be avoided.

## 2.6.3 Single-user systems

All databank file handling in single-user systems is performed by the user's own VMS process. This means that databank files created in single-user systems are owned by the VMS user.

All databases that are to be opened in single-user mode must be entered in the normal fashion into the file MIMLIB7:SQLHOSTS.DAT. The name of the file for the SYSDB databank is always SYSDB7.DBF in the directory specified in the SQLHOSTS file.

---

**Note:** It is possible to define a database in SQLHOSTS.DAT that specifies SYSSDISK:[] as the directory where the SYSDB7.DBF file is found. If this database is accessed in single-user mode, the SYSDB7.DBF file residing in the default directory defines the database. It is then possible to access several databases in single-user mode, and to switch between them with the DCL command SET DEFAULT.

---

All databank file names (except SYSDB7.DBF) are stored in the data dictionary. If full file specifications were given when the databanks were created, the locations are unambiguous. If device, directory or file type are omitted, the following default values apply:

Device:	Same as the SYSDB7.DBF file.
Directory:	Same as the SYSDB7.DBF file.
Type:	.DBF

The file name specified for a databank can be read from the data dictionary with the LIST or DESCRIBE command in ISQL or BSQL (see *MIMER/SQL Interactive User's Manual*).

#### 2.6.4 Multi-user systems

All databank file handling in multi-user systems is performed by the MIMER I/O process, so that databank files in multi-user systems are created by the MIMER system.

The location of the SYSDB databank is specified in the SQLHOSTS.DAT file, as described in the *MIMER Installation Guide for VMS*. All other databank file names are stored in the data dictionary. If full file specifications were given when the databanks were created, the locations are unambiguous. If device or directory are omitted, the default is the same as the SYSDB7 file (the SYSDB databank). The default for file type is .DBF.

The form in which the file name was specified may be read from the data dictionary with the LIST or DESCRIBE utility in ISQL and BSQL (see *MIMER/SQL Interactive User's Manual*).

---

**Note!** Users who create databanks in multi-user systems do not own the corresponding files, and these files will not be placed in the user's directory structure unless this is specifically specified. A user who creates a MIMER databank will not necessarily have access to the databank file in the operating system.

---

#### 2.6.5 Conversion between single and multi-user systems

If a database created in single-user mode is to be used in multi-user systems or vice versa, certain precautions must be observed with regard to databank file definition:

- Files created in single-user mode must be accessible for read and write by SYSTEM if they are to be used in multi-user mode, since file access is performed by the MIMER I/O process (which has SYSPRV privilege). The creator of the file should change the protection if necessary.
- Conversely, files created in multi-user mode must be accessible for read and write by the individual VMS users who need to access the databanks in single-user mode. Since multi-user databank files are created by the MIMER I/O system process, protection on these files may only be changed by users with the SYSPRV privilege in VMS.

### 2.6.6 Copying and moving databank files

Databank files may be renamed and copied to other locations in the operating system. If files are moved or copied, the file name in the data dictionary must be altered. This is done with the ALTER DATABANK statement in ISQL or BSQL (see *MIMER/SQL Reference Manual*). This statement may also be used to correct incomplete file specifications in the data dictionary, without actually changing the file location. A databank may only be altered in SQL by its creator.

Please note that once the multi-user system has opened a file, it will not close it until one of the following happens:

- The multi-user system is stopped.
- The file name of the databank or shadow is altered with the SQL statement ALTER DATABANK or ALTER SHADOW.
- The file is a shadow, and it is suspended.
- The databank or shadow is dropped. However, if it is dropped it will **also be deleted**.

While the multi-user system has the file open, it must not be copied.

---

**Note:** Databanks cannot be moved between databases by copying the databank file. The Export/Import utility must be used instead.

---

### 2.6.7 Dropping databanks or shadows

If a databank or shadow is dropped, the corresponding file will also be deleted from the disk. Please note that dropping a MIMER ident will also drop all databanks that the ident has created. When a databank is dropped all shadows of the databank will also be dropped.

## 2.7 Error codes

### 2.7.1 MRS errors

The MRS system is activated every time a MIMER module is started. This system checks the serial number of the CPU and compares it to an encoded software key to see if the installation is licensed to use the module. The software key is located in the file referenced by the logical name MIMKEY7. Every MIMER user must have read access to this file. If the MRS system finds any errors or mismatches, it will print an error message and terminate the program. See the *MIMER Installation Guide for VMS* for a further description of the MRS system.

All MRS error messages have the format:

```
MIMER-MRS Error:  error text
```

The error texts are listed and explained in the table below.

Error text	Description
Filename syntax error	The MIMKEY7 logical name does not translate to a valid file name.
File not found	The MIMKEY7 file was not found. Make sure that the MIMKEY7 logical name points to the security key file, and that the files exists.
File protection violation	The MIMKEY7 file is protected from read access. Contact the system administrator.
File locked	The file is locked for read access. Try again later.
Maximum no of opened files exceeded	The program has opened too many sequential files.
File create error	The file could not be created.
Device full	The device is full.
Diskquota exceeded	A process quota has been exceeded.
Other file error	The MIMKEY7 file could not be opened for a reason other than those previously listed. Try typing the MIMKEY7 file at the terminal.
Illegal file access option	Internal error.
Key record too long	MRS system incompatible with the key file.
Key file read error	The MIMKEY7 file could not be read, e.g. the MIMKEY7 file contains a record of more than 80 characters. Split the line into several lines.
Incorrect software key	The MIMKEY7 file contains illegal characters. Only hexadecimal characters are allowed - digits and the uppercase characters 'A' to 'F'.
Time limit exceeded	The "last date of use" has expired for the module.

Illegal CPU id	The CPU serial number does not match the software key.
Illegal module id	The installation is not authorized to use the module.
Illegal version	The installation is not authorized to use this version of the module/function.
Unauthorized use	The installation is not licensed to use the module/function.

### 2.7.2 MIMER/DB system access errors

This section describes the VMS specific error codes relevant to MIMER/DB operation. A more general consideration of error handling and a full list of MIMER error codes is given in the *MIMER/SQL Programmer's Manual*.

Execution of the current program is terminated if the error is severe.

#### Single-user systems

Only one user may operate a single-user system at any given time. If an attempt is made to start a single-user system when the MIMER system is already in use, the logon procedure will be stopped with the message:

```
MIMER/DB kernel error -16144 in function DKOPD1, 1996-07-28 13:23:30.18
  Databank SYSDB, filename SYSDB7
  File locked, OS error message:
  '%SYSTEM-W-ACCONFLICT, file access conflict'
```

```
MIMER/DB fatal error -16144 in function
  Databank SYSDB, file mimroot7:<dbank>SYSDB7 locked by other user
```

#### Multi-user systems

Attempts to run multi-user applications on a multi-user system that has not been started give the following message:

```
MIMER/DB fatal error -18901
  multi-user system not started.
```

Login may be denied with the DB error code:

```
MIMER/DB fatal error -18902
  MIMER logins are currently disabled, try again later
```

This error code appears when the system manager has disabled logins to the multi-user system (with the command MIMCONTROL/DISABLE). Logins to the multi-user system will be denied until the system manager re-enables them.

Login may also be denied with the DB error code:

```
MIMER/DB fatal error -18921 in function
  Machine dependent error -18921, refer to users guide for
  an explanation
```

This error message means that the number of users logged in to the multi-user system has reached its limit (set by the USERS parameter in the MULTIDEFS.DAT file, described in the *MIMER Installation Guide for VMS*). Logins will be denied until the number of users logged on to the multi-user system drops below the specified limit.

### 2.7.3 Direct access I/O errors

The error codes listed below may be returned from direct access I/O operations on databank files. The solution to the error situation is, in most cases, VMS-specific and the user is referred to the relevant VMS documentation, where necessary.

In most cases, when MIMER/DB receives error codes from the operating system when doing I/O, the VMS error message will be displayed.

The direct access I/O error codes are:

-114x      -124x      -214x      -1614x

where 'x' represents one of the following digits:

#### 1 Syntax error in physical file name

Invalid file name specified as physical databank name.

#### 2 File not found

No databank file with the specified name was found. Check the location of the system databanks if this error code is returned at a login attempt (particularly in single-user systems), or check the physical file name stored in the data dictionary for other databanks.

#### 3 File protection violation

The user is not allowed to create/access this databank due to VMS file protection attributes.

#### 4 File locked by other user

This databank file is currently held open by another process, and cannot be accessed.

#### 5 Too many open databanks

There is a limit of 1000 concurrently open databanks in both single-user and multi-user systems (defined in CONFIG.DAT). Close any obsolete databanks to make it possible to open a new one.

#### 6 File create error

File could not be created - device full.

#### 7 File create error

File could not be created - disk quota exceeded.

#### 8 Device not ready

Attempt to access a disk that is not ready (e.g. not mounted).

This error code is also used to indicate network problems when the remote shadowing option is used.

#### 9 Other error

Other error during databank open.

### 2.7.4 Sequential access I/O errors

The following error codes may be returned when a MIMER module tries to open a sequential access file:

- 1 Syntax error in file name
- 2 File not found
- 3 File protection violation
- 4 File locked by other process
- 5 Too many opened files  
There is a limit of 30 concurrently open files.
- 6 File create error  
File could not be created - device full.
- 7 File create error  
File could not be created - disk quota exceeded.
- 8 Device not ready
- 9 Other error

## 2.8 Developing applications

### 2.8.1 Using the old call level interface

It is generally accepted that current application development will be undertaken using ESQL (see Section 3.3), however, an old call level interface to MIMER functionality exists and Sections 2.8.1.1 and 2.8.1.2 below refer to this.

#### 2.8.1.1 Writing applications in C

The MIMER routines require that all arguments are passed by reference. This means that whenever an integer argument is passed to a MIMER routine, a pointer to an integer variable must be used.

All MIMER string arguments (specified as C or 'ref') are compatible with a normal char\* type in C (pointer to character). MIMER does **not** use null-terminated strings. The length of the argument is determined by other means (fixed-length or space-terminated - please read the appropriate documentation).

#### 2.8.1.2 String parameters in FORTRAN

Some MIMER routines specify character string parameters (these arguments are specified as "Cx"). The actual values given as parameters to such routines must be compatible with FORTRAN-66 Hollerith strings. Variables with the FORTRAN-77 CHARACTER data type may not be used.

**VAX**

Character string constants may be used, delimited by apostrophes ('), but Hollerith strings (following the FORTRAN-66 standard) are generally to be preferred for reasons of portability. ♦

---

**Note!** The use of Hollerith strings for character constants does not guarantee portability of FORTRAN programs to other computer systems. Machine-specific differences in FORTRAN dialects must be accounted for in application programs designed to be run on different machines.

---

## Examples

The following formulations for a call to the OPFLF2 routine (open source form library) are all valid. Preferable variants are shown in bold type:

---

```
INTEGER RC
..
CALL OPFLF2(RC,6HHOTEL )
```

---

```
INTEGER RC, TABLE(2)
DATA TABLE/4HHOTE,2HL /
..
CALL OPFLF2(RC, TABLE)
```

---

```
INTEGER RC
CHARACTER*6 TABLE/'HOTEL '/
..
CALL OPFLF2(RC,%REF(TABLE))
```

---

**VAX**

The example that follows does *not* work on Alpha/VMS:

---

```
INTEGER RC
..
CALL OPFLF2(RC,'HOTEL ') ◆
```

---

The example below is *not* acceptable:

```
INTEGER RC
CHARACTER*6 TABLE/'HOTEL '/
..
CALL OPFLF2(RC, TABLE)
```

## 2.8.2 Optimizing compilers

Some MIMER/DB application programs may give incorrect results under certain circumstances if the object code is optimized by the compiler. The problem originates mainly from loop-optimization where variables used inside the loop are bound to columns or variables inside MIMER/DB. Such binding occurs either when using host variables in embedded SQL applications, or when calling the PROJE2 and/or SELEC2 routines. In such cases, the compiler may choose to put the variable in a register for the duration of the loop. When MIMER/DB changes (or reads) the actual value in memory, the register version of the variable will be incorrect.

To avoid this problem, all variables bound to columns or variables in MIMER/DB should be declared as VOLATILE. This will direct the compiler that the variable should not be allocated in a register and is expected to change at any time.

### 2.8.3 Linking applications

Most MIMER applications require one of the MIMER/DB libraries when linking (either single-user or multi-user mode). Since these libraries are shared images in MIMER version 7, an option (.OPT) file is needed. The standard MIMER installation provides four such option files; all can be found in the MIMLIB7: directory. Each of the option files has a logical name pointing to it:

Logical Name	Translation	Usage
MIMERM7	MIMLIB7:MIMERM.OPT	MIMER/DB multi-user mode with MIMER/FM
MIMERS7	MIMLIB7:MIMERS.OPT	MIMER/DB single-user mode with MIMER/FM
MIMERMSH7	MIMLIB7:MIMERMSH.OPT	MIMER/DB multi-user mode with MIMER/SH
MIMERS7SH7	MIMLIB7:MIMERS7SH.OPT	MIMER/DB single-user mode with MIMER/SH

In addition to the MIMER/DB and FM runtime libraries, the option files listed above also include utility libraries (LRU and LR). Other MIMER runtime libraries (such as RGLIB or PGLIB) are not included in the option files. Instructions on how to use the other runtime libraries are given in Chapter 3.

## 2.9 Using alternative MIMER installations

The MIMSETUP7 command procedure is found in the file [MIMER7.LIBRARY] MIMSETUP7.COM. It defines the logical names needed to run MIMER applications. If the user specifies GROUP- or SYSTEM-wide definition, the procedure will also install some shareable libraries. This is further described in the *MIMER Installation Guide for VMS*.

This command procedure can be used by ordinary MIMER users if they want to use an alternative MIMER installation. The command procedure can override the default logical name definitions (in the SYSTEM logical name table) by defining the logical names in a user's PROCESS table.

### Syntax:

```
$ @disk:[MIMER7.LIBRARY]MIMSETUP7 mimroot [lnmtable]
```

**mimroot** Directory specification giving the MIMER runtime directory tree.

**lnm table** Specifies which logical name table to use when defining the logical names needed to access a MIMER installation. Valid values: SYSTEM, GROUP, JOB or PROCESS. Default: PROCESS.

### Example:

The following command shows how any user can override the default definition of the logical names. This is useful when a user wishes to use an alternate MIMER installation. No shareable images are installed.

```
$ @MIMLIB7:MIMSETUP7 otherdisk:[OTHERMIMER]
```

## 2.10 Converting a version 7.1 or version 7.2 database

MIMER version 7.3 includes some new dictionary views. Before a version 7.1 or a version 7.2 database can be used in a version 7.3 environment, some dictionary tables must be reloaded. This is done by the UPGR73 program. Please see the *MIMER Installation Guide for VMS* for more information.

Programs linked with MIMER version 7.1 or 7.2 do not have to be relinked. When version 7.3 is properly installed and set up, the MIMER/DB shareable libraries are replaced. The old applications will use the new libraries automatically.

## 2.11 Converting a version 5 database

**VAX**

Note that the following discussion about the conversion of version 5 databases applies to VAX/VMS only. If you are not using VAX/VMS, you should skip the rest of this section.

The MIMER version 7 databank format differs from the MIMER version 5 format. A version 5 database must be converted before they can be used in a MIMER version 7 environment.

### 2.11.1 Upgrading databanks

To use a MIMER version 5 database with MIMER/DB version 7.3, you must convert the database with the MIMEXE7:CONV57 program. This program is further described in the *MIMER Installation Guide for VMS*.

---

**Note:** Once the CONV57 program has converted the database, you cannot use the database with MIMER version 5.

---

When the database has been converted, it must be given a database name. The database name is entered together with the location of the SYSDB7.DBF file in the MIMLIB7:SQLHOSTS.DAT file. The SQLHOSTS file is described in the *MIMER Installation Guide for VMS*.

### 2.11.2 Using version 5 applications with MIMER version 7

Most user-written applications can be used directly in the MIMER version 7 environment by redefining two logical names:

```
$ DEFINE MIMDB5M MIMDB7M
$ DEFINE MIMDB5S MIMDB7S
```

When these names are defined, all applications that were linked with MIMER/DB version 5 will now run in the version 7 environment. You don't have to relink the applications if you use this method.

If you want to use the new run-time libraries for other modules, such as FMR, SHR or PGR, you must relink your applications. ♦



## 3 MODULE-SPECIFIC INFORMATION

Version number 7.3 applies for all modules unless otherwise specified. Where older versions are used, these are fully compatible with all features of version 7.3.

### 3.1 MIMER/CONV

VAX

The MIMER/CONV module is not available in the Alpha/VMS implementation of MIMER since MIMER V4 was never implemented on Alpha/VMS.

MIMER/CONV is the module supplied on VAX/VMS for conversion of version 4 databases to version 7. The facility is documented in the separate manuals *MIMER Database Migration* and *MIMER Application Migration*.

MIMER/CONV is only available on sites with an established MIMER version 4 environment.

#### 3.1.1 Accessible files

The CONV module contains the following files accessible to the user:

MIMEXE7:CONV47	The CONV47 program, for converting version 4 databases.
MIMEXE7:SHCONV47	The SHCONV47 program, for preparing version 4 SYSSH for conversion to version 7.

#### 3.1.2 Running the conversion

The CONV47 and SHCONV47 programs operate on version 4 databases in single-user mode only. Please read the database migration documentation for more details.

The CONV47 program generates sequential files containing definitions and data for the version 4 database. If desired, these generated files can be directed to tape instead of to disk. Refer to Appendix A for details.

The version 7 environment is established from the sequential files generated by CONV47 by using the Export/Import function in the system utilities (MIMER/UTIL).

CONV47 does not modify the version 4 databanks. When the conversion programs have been executed, the version 4 applications may resume operations again.

### **3.1.3 File type conventions**

The CONV47 program will generate one definition file (containing data dictionary definitions) and one or more data files (containing the data in the tables). These files have the default file type '.EXP'.

The CONV47 program will also create a log file. This file will have the default extension '.CNV'.

The SHCONV47 program does not use sequential files. ♦

## 3.2 MIMER/DB

This module includes the relational database manager, including a multi-user system, client/server facilities, SQL compiler, etc. All database handling in all other modules is done through MIMER/DB.

### 3.2.1 Accessible files in MIMER/DB

The DB module includes the following files that are accessible to users:

MIMEXE7:CONV57.EXE	Program that converts a MIMER version 5 database to MIMER version 7.
MIMEXE7:DBC.EXE	Database check program. Can be used to check if a databank is physically damaged.
MIMLIB7:LR.OLB	MIMER Library Routines.
MIMLIB7:LRU.OLB	MIMER Library Routines Upper level.
MIMLIB7:MIMDB7M.EXE	DB multi-user runtime library.
MIMLIB7:MIMDB7S.EXE	DB single-user runtime library.
MIMLIB7:MIMERM.OPT	Option file to link multi-user programs with MIMER/FM.
MIMLIB7:MIMERMSH.OPT	Option file to link multi-user programs with MIMER/SH.
MIMLIB7:MIMERS.OPT	Option file to link single-user programs with MIMER/FM.
MIMLIB7:MIMERSSSH.OPT	Option file to link single-user programs with MIMER/SH.
MIMEXE7:IXCHECKS.EXE	Program to check secondary indexes in single-user mode.
MIMEXE7:SDBGEN.EXE	System databank generation program.
MIMEXE7:UPGR73.EXE	Utility for upgrade the database to V7.3.

### 3.2.2 The MIMER/DB runtime libraries

The MIMER/DB runtime libraries (single and multi-user mode) are shareable images. Shareable images provide the following advantages:

- Sizes for the executable (.EXE) application files are significantly reduced.
- Linking a MIMER/DB application is faster, since all internal references in DB are already resolved.
- The MIMER/DB code is shared if the image is installed. This results in reduced main memory usage and enhanced performance.

An option (.OPT) file should be used to link an application to a shareable image. The MIMER/DB module includes four such files (see previous section). Each of the files can be referenced by a logical name (see Section 2.1.2 for a description of logical names). By using the option files, a MIMER/DB application can be linked to a shareable image with the command:

```
$ LINK application,MIMERM7/OPT
```

### 3.3 MIMER/ESQL

The ESQL (Embedded SQL) module provides the ability to embed SQL statements in an ordinary programming language (FORTRAN, COBOL or C). By the use of the preprocessor provided in this module, the embedded program is converted to a pure host-language program that can be compiled by the ordinary language compiler.

#### 3.3.1 Accessible files

The ESQL module includes the following files that are accessible to users:

MIMEXE7:ESQL.EXE	ESQL preprocessor
MIMLIB7:ESQL.CLD	ESQL command definitions.
MIMLIB7:EXAMPLE.EC	Simple ESQL/C example program.
MIMLIB7:EXAMPLE.ECO	Simple ESQL/COBOL example program.
MIMLIB7:EXAMPLE.EFO	Simple ESQL/FORTRAN example program.
MIMLIB7:BLOBSAMP.C	Example program demonstrating how large binary objects can be stored in a database.
MIMLIB7:DSQL.EC	Example program library showing the use of embedded dynamic SQL in the C language.
MIMLIB7:DSQL.H	Header file for DSQL.EC.
MIMLIB7:DSQLSAMP.C	Source code for interactive SQL query program using the DSQL.EC routines.
SY\$LIBRARY:MIMESQL.H	Header file containing type definitions.

#### 3.3.2 VAX/VMS compilers supported

The output generated by the ESQL preprocessor can be compiled by one of the language preprocessors listed below. Other language preprocessors (compilers) from other software distributors may or may not be able to compile the ESQL output. MIMER does not support any compilers other than those listed.

**VAX**

Supported compilers on VAX systems:

VAX/VMS C	sold by Digital Equipment
VAX/VMS FORTRAN	" " " "
VAX/VMS COBOL	" " " "

When using COBOL, the source program must be formatted according to the ANSI rules. Use the /ANSI switch when compiling the resulting COBOL program. ◆

**AXP**

On Alpha/VMS, the following compilers are supported:

DEC C	sold by Digital Equipment
DEC FORTRAN	" " " " ◆

### 3.3.3 ESQL command syntax

The ESQL preprocessors are started with the DCL command ESQL, which must be defined in the terminal session to make the preprocessors available (unless the installer has done it system-wide). To define the ESQL command, execute the following DCL command:

```
$ SET COMMAND MIMLIB7:ESQL
```

The ESQL command which starts the preprocessor has the following syntax:

```
$ ESQL/<language> [/OUTPUT=<outfile>] <infile>
```

**<language>** is one of FORTRAN, COBOL or C. This switch is required.

**<outfile>** specifies where the result of the preprocessor should be stored. If the /OUTPUT qualifier is not given, the output file will have the same name as the input file, but with a different file type (as listed below).

**<infile>** gives the name of the source file.

**/SILENT** If the /SILENT switch is specified, ESQL will not report syntactical errors on SYS\$OUTPUT. The default is /NOSILENT.

### 3.3.4 File handling conventions

The default file extensions for input and output files depend on the host language used:

Language	Input	Output
C	.EC	.C
COBOL	.ECO	.COB
FORTRAN	.EFO	.FOR

### 3.3.5 Type definitions

For ESQL/C, the file SYS\$LIBRARY:MIMESQL.H contains type definitions used in generated C code.

### 3.3.6 Building the simple EXAMPLE program

The source code of a simple example program is included in the ESQL distribution. The program logs in to the default database using username SYSADM, password SYSADM (change the program as needed). The program will then display the contents of the standard dictionary view INFORMATION\_SCHEMA.TABLES.

There are three versions of the program; one for each supported host language. The programs are written according to international SQL standards and should be able to use any standard-compliant database without modification.

The following example illustrates how EXAMPLE.EFO is compiled, linked and run. The programs written in the other languages can be linked together in a similar manner.

```
$ ESQL/FORTRAN MIMLIB7:EXAMPLE           ! Preprocess source code
$ FORTRAN EXAMPLE                         ! Compile preprocessed code
$ LINK EXAMPLE,MIMLIB7:MIMERM/OPT        ! Link an executable program
$ RUN EXAMPLE                             ! Run it
```

### 3.3.7 Building the DSQLSAMP example program

The file MIMLIB7:DSQL.EC demonstrates how a C program can be constructed that uses dynamic SQL (passing SQL statements constructed at runtime to the database manager for execution). The DSQL.EC file uses dynamic SQL syntax according to international SQL standards. The DSQL.EC file contains a collection of routines defines a simple but convenient API (Application Programming Interface) for dynamic SQL that can be used from other C programs.

The file MIMLIB7:DSQLSAMP.C contains a program that calls the routines in DSQL.EC. The DSQLSAMP program allows the user to enter a SQL statement that will be executed directly (similar to the BSQL program).

To build the DSQLSAMP example program, the following commands can be used:

```
$ ESQL/C MIMLIB7:DSQL                     ! Preprocess DSQL source code
$ COPY MIMLIB7:DSQL.H SYS$DISK:[ ]       ! Get header file
$ COPY MIMLIB7:DSQLSAMP.C SYS$DISK:[ ]   ! Get main program
$ CC DSQL                                 ! Compile preprocessed code
$ CC DSQLSAMP                             ! Compile sample program
$ LINK DSQLSAMP,DSQL,MIMLIB7:MIMERM/OPT ! Link executable program
$ RUN DSQLSAMP                             ! Run it
```

## 3.4 MIMER/FMD

The FMD (Forms Manager Development) module contains a forms editor that allows users to create screen forms. It also contains a conversion utility that can convert forms from the older screen interface SH to FM forms.

Version 5.2.1 of MIMER/FMD is used with MIMER version 7.3.

### 3.4.1 Accessible files

The FMD module includes the following files that are accessible to users:

MIMEXE7:FMEM.EXE	FM editor multi-user mode.
MIMEXE7:FMES.EXE	FM editor single-user mode.
MIMEXE7:FMSHCM.EXE	FM/SH compiler multi-user mode.
MIMEXE7:FMSHCS.EXE	FM/SH compiler single-user mode.

---

**Note:** The installation process does not require that all of the above files are generated. For example, the system installer may have chosen to generate only some of these files to save disk space. Users should contact the system administrator if a file they need is missing.

---

### 3.4.2 File handling conventions

Forms may be documented in external sequential files with the PRINT function in the FM editor. One file is generated for each form printed. The file name is the same as the form name, with file type .MPR.

---

**Note!** Names of forms to be printed with the PRINT function may only contain characters permitted in VMS file names.

---

### 3.4.3 Function keys in the FM editor

#### KEYPAD USAGE

PF11 <i>PF1</i>	PF12 <i>PF2</i>	PF13 <i>PF3</i>	PF14 <i>PF4</i>
PF7 7	PF8 8	PF9 9	CLEAR /
PF4 4	PF5 5	PF6 6	DEL CHAR ,
PF1 1	PF2 2	PF3 3	SEND/ ENTER
PF10 0		INS ↔ OVER .	

**KEY FUNCTIONS**

<b>KEY</b>	<b>FUNCTION</b>
PF1...PF14	Programmed function keys
CLEAR	Clear to the end of field
DEL CHAR	Delete current character
INS/OVER	Set/reset insert mode (toggle)
SEND/ENTER	Send (end of input)
DEL	Delete preceding character
BACKSPACE	Jump to previous field
TAB	Jump to next field
RETURN	Jump to the first field in the next line. If used from the last field in a form, RETURN is interpreted as SEND.
CTRL A	Set/reset insert mode (toggle)
CTRL D	Delete character
CTRL E	Jump to end of field
CTRL K	Clear to end of field
CTRL N	Toggle application / numeric keypad
CTRL P	Print screen
CTRL R	Refresh screen
CTRL W	Refresh screen

## 3.5 MIMER/FMR

This module includes the runtime libraries for the Forms Manager, providing similar functions in both synchronous and asynchronous environments.

### 3.5.1 Accessible files

The FMR module includes the following files that are accessible to users:

MIMLIB7:FMLIB.OLB                      FM runtime library.  
 MIMLIB7:FMSHI.OLB                      SH runtime library interface to FM.

### 3.5.2 Supported terminal types

The MIMER/FM module uses the VMS SMG facility to communicate with terminals. Because of this, MIMER/FM will support any terminal that SMG supports. To inform SMG of the terminal type being used, the user must use the DCL command SET TERMINAL.

#### Examples:

```
$ SET TERM/VT100        ! Select VT100 terminal type
$ SET TERM/INQ         ! Inquire the terminal for type.
                          This will only work on terminals supported
                          by Digital Equipment Corporation.
```

### 3.5.3 Function keys in MIMER/FMR

#### KEYPAD USAGE

PF11 <i>PF1</i>	PF12 <i>PF2</i>	PF13 <i>PF3</i>	PF14 <i>PF4</i>
PF7 7	PF8 8	PF9 9	CLEAR /
PF4 4	PF5 5	PF6 6	DEL CHAR ,
PF1 1	PF2 2	PF3 3	SEND/ ENTER
PF10 0		INS ↔ OVER .	

**KEY FUNCTIONS**

<b>KEY</b>	<b>FUNCTION</b>
ARROW DOWN	Move cursor down
ARROW UP	Move cursor up
ARROW LEFT	Move cursor left
ARROW RIGHT	Move cursor right
PF1...PF14	Programmed function keys
CLEAR	Clear to the end of field
DEL CHAR	Delete current character
INS/OVER	Set/reset insert mode (toggle)
SEND/ENTER	Send (end of input)
DEL	Delete preceding character
BACKSPACE	Jump to previous field
TAB	Jump to next field
RETURN	Jump to the first field in the next line. If used from the last field in a form, RETURN is interpreted as SEND.
CTRL A	Set/reset insert mode (toggle)
CTRL D	Delete character
CTRL E	Jump to end of field
CTRL K	Clear to end of field
CTRL N	Toggle application / numeric keypad
CTRL P	Print screen
CTRL R	Refresh screen
CTRL W	Refresh screen

On VT200 and VT300 keyboards, 9 additional undefined PF-keys are available.

The top row of keys on the keyboard has the following key assignments (Prv Fld = Previous Field):

F6	F7	F8	F9	F10	F11	F12	F13	F14
					ESC	BS	LF	
PF15	PF16	PF17	PF18	PF19		Prv Fld	SEND	PF20
					F17	F18	F19	F20
HELP	EXECUTE				PF21	PF22	PF23	PF24

---

**Note.** As a feature of VMS keyboard software, PF15 (F6 on VT200/VT300 terminals) generates an interrupt in line editing mode. To use PF15 in MIMER/FM environments, the DCL command:

```
$ SET TERM/NOLINE
```

must be issued before starting the FM application. It is better to avoid using PF15 in FM applications if this is possible.

---

### 3.5.4 Set keypad mode

The keypad can be set dynamically from the application program for numeric or application keypad mode with the SKPMF2 command. Note that this routine is supported in the VMS version only. It should not be used in programs that are to be portable.

#### Syntax:

```
SKPMF2 (MODE)
```

MODE

Integer input parameter specifying the mode to be set. If mode=0, the keypad is set to numeric mode. If mode=1, the keypad is set to application mode. The default mode after a call to INITF2 is 1 (application mode).

Any value for mode other than 0 or 1 is treated as 1 and sets application mode.

### 3.5.5 Linking applications to MIMER/FMR

An application using MIMER/FMR should use one of the option files described in Section 2.10.4. These option files automatically include MIMER/DB and some utility libraries. A MIMER/FMR application is linked with the command:

```
$ LINK application,MIMERM7/OPT
```

The SHFM-interface is a library that maps all MIMER/SHR routines to MIMER/FMR functions. Most (but not all) of the functions in MIMER/SHR are mapped. To use this interface, the FMSHI object library must be included in the link specification:

```
$ LINK application,MIMLIB7:FMSHI/LIB,MIMERM7/OPT
```

## 3.6 MIMER/ISQL

The ISQL (Interactive SQL) module provides the ability to enter SQL statements into a full screen editor and execute the statements interactively. BSQL (Batch SQL) allows the user to execute SQL commands read from an input file (or the terminal).

### 3.6.1 Accessible files in MIMER/ISQL

The ISQL module includes the following files that are accessible to users:

MIMEXE7:BSQLM.EXE	Batch SQL program (scrolling interface) in multi-user mode.
MIMEXE7:BSQLS.EXE	Batch SQL program (scrolling interface) in single-user mode.
MIMEXE7:ISQLM.EXE	Interactive SQL program (forms driven) in multi-user mode.
MIMEXE7:ISQLS.EXE	Interactive SQL program (forms driven) in single-user mode.
MIMLIB7:CREHOTDB.DAT	SQL commands to create the hotel database environment.
MIMLIB7:EXAMPLES.DAT	SQL statement examples.

---

**Note:** The installation process does not require that all of the above files are generated. For example, the system installer may have chosen to generate only some of these files to save disk space. Users should contact the system administrator if a file they need is missing.

---

### 3.6.2 Executing DCL commands

Both BSQL and ISQL have a command named "COMMAND". The command will spawn a DCL process to execute the DCL command given as parameter to the "COMMAND" command. If no parameter is given, the user's terminal will be connected to the spawned process. To leave the spawned process and return to the ISQL/BSQL session, use the DCL command LOGOUT.

### 3.6.3 Keyboard interrupt

BSQL allows the user to interrupt the execution of commands by pressing CTRL-C. This will take the user to the BSQL prompt.

---

**Note!** CTRL-Y will interrupt the entire program and bring the user back to the DCL level. The interrupted program can be restarted at the point of interruption if the command CONTINUE is given immediately after the interrupt.

---

### 3.6.4 Function keys in the ISQL full screen editor

#### KEYPAD USAGE

TOGGLE <i>PF1</i>	HELP <i>PF2</i>	RESET <i>PF3</i>	EXIT <i>PF4</i>
SPLIT C 7	UP 15 8	JOIN C 9	CLEAR /
LEFT 80 4	UNDO 5	RIGHT 80 6	DEL CHAR ,
REMOVE 1	DOWN 15 2	EOL 3	SEND/ ENTER
EXECUTE 0		INS ↔ OVER .	

#### KEY FUNCTIONS

KEY	FUNCTION
PF1...PF14	Programmed function keys
CLEAR	Clear to the end of field
DEL CHAR	Delete current character
INS/OVER	Set/reset insert mode (toggle)
SEND/ENTER	Send (end of input)
DEL	Delete preceding character
BACKSPACE	Jump to previous field
TAB	Jump to next field
RETURN	Jump to the first field in the next line. If used from the last field in a form, RETURN is interpreted as SEND.

KEY	FUNCTION
CTRL A	Set/reset insert mode (toggle)
CTRL D	Delete character
CTRL E	Jump to end of field
CTRL K	Clear to end of field
CTRL N	Toggle application / numeric keypad
CTRL P	Print screen
CTRL R	Refresh screen
CTRL W	Refresh screen

On VT200 and VT300 keyboards, 9 additional undefined PF-keys are available.

The top row of keys on the keyboard has the following key assignments:

F6	F7	F8	F9	F10	F11	F12	F13	F14
					ESC	BS	LF	

PF15	PF16	PF17	PF18	PF19		Prv Fld	SEND	PF20
------	------	------	------	------	--	------------	------	------

				F17	F18	F19	F20
--	--	--	--	-----	-----	-----	-----

HELP	EXECUTE	PF21	PF22	PF23	PF24
------	---------	------	------	------	------

---

**Note:** As a feature of VAX/VMS keyboard software, the function key PF15 (F6 on VT200/VT300 terminals) generates an interrupt in line editing mode. To be able to use PF15 in the ISQL editor, issue the DCL command:

```
$ SET TERM/NOLINE
```

before starting ISQL.

---

### 3.6.5 The MIMINI initiation file and ISQL/BSQL

The initialization file MIMINI may be used to store login procedures for both ISQL and BSQL. The default name for the file is MIMINI.DAT in the current default device and directory. The initialization file may also be referenced by the logical name MIMINI.

## 3.7 MIMER/PGD

The PGD (Program Generator Development) module provides 4GL capabilities. It contains functions to load and develop PG programs. The programs may be run interactively. When the programs have been tested, FORTRAN or COBOL code can be generated from the PG application source code to give the performance needed for production use.

### 3.7.1 Accessible files

The PGD module includes the following files that are accessible to users:

MIMEXE7:PGM.EXE	PG using MIMER/FM for screen I/O in multi-user mode.
MIMEXE7:PGS.EXE	PG using MIMER/FM for screen I/O in single-user mode.
MIMEXE7:PGSHM.EXE	PG using MIMER/SH for screen I/O in multi-user mode.
MIMEXE7:PGSHS.EXE	PG using MIMER/SH for screen I/O in single-user mode.

---

**Note:** The installation process does not require that all of the above files are generated. For example, the system installer may have chosen to generate only some of these files to save disk space. Users should contact the system administrator if a file they need is missing.

---

The code generator files COBGEN.EXE and FORGEN.EXE in MIMEXE7: should also be available to PGD users.

### 3.7.2 File handling conventions

#### Input

Sequential files containing PG commands and program, procedure and report definitions are loaded by the LOAD FILE command with a default file type of .PG if none is specified.

#### Output

Output files created by the OUTPUT command are given the file type .DAT if none is explicitly specified.

Output files created by the GENERATE command are given the file type .FOR or .COB for FORTRAN and COBOL respectively.

Application program source code files generated from PG have the same name as the PG program.

---

**Note!** The PG program name may not contain any special characters forbidden in VMS file names. If unacceptable characters are used in a program or procedure name, the generation will fail.

---

### 3.7.3 Executing DCL commands

The PG program has a command named "COMMAND". This command will spawn a DCL process to execute the DCL command given as the parameter to the "COMMAND" command. If no parameter is given, the user's terminal will be connected to the spawned process. Leave the spawned process and return to the PG session with the DCL command LOGOUT.

### 3.7.4 Keyboard interrupt

PG allows the user to interrupt executing commands by pressing CTRL-C. This will take the user to the PG> prompt.

---

**Note!** CTRL-Y will interrupt the entire program and bring the user back to the DCL level. At this point, the interrupted program can be restarted at the point of interruption if the command CONTINUE is given immediately after the interrupt.

---

### 3.7.5 The MIMINI initiation file and PG

The initialization file MIMINI may be used to store login procedures which will be executed automatically at the start of a PG session. The default name for the file is MIMINI.DAT in the current device and directory. The initialization file may also be addressed by the logical name MIMINI.

Commands to be executed automatically at the beginning of a PG session may be written in a PG program called PGINIT and stored in the user's default procedure library as described in the *MIMER/PG Reference Manual*.

### 3.7.6 Code generators

When code is generated from PG, a subprocess will be created. This subprocess will execute one of the code generator programs (MIMEXE7:FORGEN, MIMEXE7:COBGEN or MIMEXE7:PCGEN). Normally, PG will wait for the completion of the subprocess before a new PG command is accepted.

Please note that if an alternate MIMER installation has been defined by the MIMSETUP7 command procedure (described in Section 2.11), the definition should be at least JOB-wide. If a PROCESS-wide definition is made, the wrong code generator program will be started.

## 3.8 MIMER/PGR

This module includes the runtime libraries needed to link PG (Program Generator) applications.

### 3.8.1 Accessible files

The PGR module includes the following files that are accessible to users:

MIMLIB7:PGFML.OLB	PG interface to MIMER/FM.
MIMLIB7:PGL.OLB	PG runtime library.
MIMLIB7:PGRUN.COM	Command procedure to link simple PG applications.
MIMLIB7:PGSHL.OLB	PG interface to MIMER/SH.

### 3.8.2 Linking a PG application

In addition to the PG runtime library (PGL.OLB) there are two PG screen interface libraries: the PGFML library, which uses the MIMER/FM interface for screen I/O, and the PGSHL library which uses the MIMER/SH interface for screen I/O. The library is selected depending on which screen I/O interface is being used. If screen I/O is not used in the application, the PGFML library should be used.

The following command links an application that uses the MIMER/FM screen interface (or does not use screen I/O at all) in multi-user mode:

```
$ LINK application,MIMLIB7:PGFML/LIB,PGL/LIB,MIMERM7/OPT
```

The following command links an application that uses the MIMER/SH screen interface in multi-user mode:

```
$ LINK application,coupling-  
file,MIMLIB7:PGSHL/LIB,PGL/LIB,MIMERMSH7/OPT
```

Note that the coupling file generated by the SH compiler should be included before entering the PGSHL library in the link list. Coupling files are described in the *MIMER/SH Reference Manual*.

### 3.8.3 The PGRUN command procedure

In the MIMLIB7 directory, there is a command procedure file named PGRUN.COM. This command procedure can be used to compile and link simple PG applications. Complicated applications (having several separately compiled procedures and/or using external libraries) cannot be linked by PGRUN. When complicated applications are to be compiled and linked, the user should write his own compile/link procedures.

**Example:**

In the example listed below, PGRUN is used to compile and link a PG-generated application named APPL. The PG tool has been instructed to generate COBOL code:

```

$ @MIMLIB7:PGRUN
Language (Fortran,Cobol)          [Fortran] : COBOL
File generated by MIMER/PG        [MAIN]   : APPL
Single or multi user mode (S,M)   [M]     : <RETURN>
MIMER/FM or MIMER/SH (FM,SH)     [FM]    : <RETURN>
$ COBOL APPL
$ LINK APPL,MIMLIB7:PGL/LIB,PGFML/LIB,MIMERM7/OPT

```

This command procedure first asks for the language that the application was generated to (COBOL in the example), the name of the generated application file (APPL), if MIMER/DB should be used in single-user or multi-user mode (default: multi-user) and the screen I/O interface to be used (default: FM). If screen I/O is not used, choose the FM option to avoid the prompt for an SH coupling file. If the RETURN key is pressed, the default value shown inside the brackets will be used. The command procedure will echo all compile and link commands that it executes on the screen. When the procedure is completed, the application can be executed.

```

$ RUN APPL

```

## 3.9 MIMER/PI

MIMER/PI is the 'level 4' programming interface that was provided with MIMER version 4. It is included in version 7 for backward compatibility reasons only, and should not be used for development of new application programs.

The PI module also contains the CL (Command Language) program, for compatibility with CL command files written for version 4 of MIMER.

### 3.9.1 Accessible files

The PI module contains the following files accessible to the user:

MIMEXE7:CLM.EXE	CL program for multi-user mode.
MIMEXE7:CLS.EXE	CL program for single-user mode.
MIMLIB7:PI.OLB	PI interface runtime library.

### 3.9.2 Linking an application that uses MIMER/PI

To link an application that uses the PI interface, the PI runtime library must be included in addition to the normal libraries specified in the standard option files:

```
$ LINK application,MIMLIB7:PI/LIB,MIMERM7/OPT
```

---

**Note:** Although the PI library is functionally equivalent to the version 4 PI library from the user's viewpoint, the version 4 or version 5 distribution of the library **cannot** be used to link applications in the version 7 environment.

---

### 3.9.3 Exit routines

Version 7 of MIMER/PI supports the PI (level 4) exit routines EXITA4 and EXITC4, as documented in the *MIMER/DB version 4 Reference Manual*.

## 3.10 MIMER/QF

The QF (Query by Forms) module allows users to perform simple table operations (find, update, insert) using a forms-driven interface.

### 3.10.1 Accessible files

The QF module includes the following files that are accessible to users:

MIMEXE7:QFS.EXE                      Query by Forms program - single-user mode.

MIMEXE7:QFM.EXE                      Query by Forms program - multi-user mode.

---

**Note:** The installation process does not require that all of the above files are generated. The system installer may have chosen to generate only some of these files to save disk space, for example. Users should contact the system administrator if a file they need is missing.

---

### 3.10.2 Function keys in MIMER/QF

#### KEYPAD USAGE

PF1 <i>PF1</i>	PF2 <i>PF2</i>	PF3 <i>PF3</i>	PF4 <i>PF4</i>
7	8	9	CLEAR /
4	5	6	DEL CHAR ,
1	2	3	SEND/ ENTER
	0	INS ↔ OVER .	

**KEY FUNCTIONS**

<b>KEY</b>	<b>FUNCTION</b>
PF1...PF14	Programmed function keys
CLEAR	Clear to the end of field
DEL CHAR	Delete current character
INS/OVER	Set/reset insert mode (toggle)
SEND/ENTER	Send (end of input)
DEL	Delete preceding character
BACKSPACE	Jump to previous field
TAB	Jump to next field
RETURN	Jump to the first field in the next line. If used from the last field in a form, RETURN is interpreted as SEND
CTRL A	Set/reset insert mode (toggle)
CTRL D	Delete character
CTRL E	Jump to end of field
CTRL K	Clear to end of field
CTRL N	Toggle application / numeric keypad
CTRL P	Print screen
CTRL R	Refresh screen
CTRL W	Refresh screen

## 3.11 MIMER/QL

The QL (Query Language) module gives the user the ability to manipulate tables by using QUEL commands. This module is mainly intended to provide compatibility with previous versions of MIMER.

### 3.11.1 Accessible files

The QL module includes the following files that are accessible to users:

MIMEXE7:QLM.EXE	Query Language program - multi-user mode. If QL procedures use screen I/O, MIMER/FM is used.
MIMEXE7:QLS.EXE	Query Language program - single-user mode. If QL procedures use screen I/O, MIMER/FM is used.
MIMEXE7:QLSHM.EXE	Query Language program - multi-user mode. If QL procedures use screen I/O, MIMER/SH is used.
MIMEXE7:QLSHS.EXE	Query Language program - single-user mode. If QL procedures use screen I/O, MIMER/SH is used.
MIMLIB7:QLLIB.OLB	MIMER/QL runtime library.

---

**Note:** The installation process does not require that all of the above files are generated. For example, the system installer may have chosen to generate only some of these files to save disk space. Users should contact the system administrator if a file that they need is missing.

---

### 3.11.2 Executing DCL commands

The QL program has a command named "COMMAND". This command will spawn a DCL process to execute the DCL command given as the parameter to the "COMMAND" command. If no parameter is given, the user's terminal will be connected to the spawned process. Leave the spawned process and return to the QL session with the DCL command LOGOUT.

### 3.11.3 Keyboard interrupt

QL allows the user to interrupt executing commands by pressing CTRL-C. This will take the user to the QL> prompt.

---

**Note!** CTRL-Y will interrupt the entire program and bring the user back to the DCL level. The interrupted program can be restarted at the point of interruption if the command CONTINUE is given immediately after the interrupt.

---

#### **3.11.4 The MIMINI initiation file and QL**

The initialization file MIMINI may be used to store login procedures and QL commands which will be executed automatically at the start of a QL session. The default name for the file is MIMINI.DAT in the current device and directory. The initialization file may also be addressed by the logical name MIMINI.

## 3.12 MIMER/RG

The RG (Report Generator) module allows the user to generate reports directly from tables using the RGF (Report Generation by Forms) utility. By using RGL (Report Generator Language) the user can construct complex reports with a very high degree of control of the layout. The RGF utility can generate RGL code.

Version 5.2.2 of MIMER/RG is used with MIMER version 7.3.

### 3.12.1 Accessible files

The RG module includes the following files that are accessible to users:

MIMEXE7:RGBATM.EXE	RG batch (scrolling interface) program in multi-user mode.
MIMEXE7:RGBATS.EXE	RG batch (scrolling interface) program in single-user mode.
MIMEXE7:RGM.EXE	RG interactive (forms driven) program in multi-user mode.
MIMEXE7:RGS.EXE	RG interactive (forms driven) program in single-user mode.
MIMLIB7:RGLIB.OLB	RG runtime library for user RG applications.
MIMLIB7:RGM.OBJ	Full screen routines for MIMER/RG.

---

**Note:** The installation process does not require that all of the above files are generated. The system installer may have chosen to generate only some of these files to save disk space, for example. Users should contact the system administrator if a file they need is missing.

---

### 3.12.2 Executing DCL commands

The RG program has a facility to execute DCL commands. When this is done the program will spawn a DCL process to execute the DCL command given as parameter. If no command is given, the user's terminal will be attached to the spawned process. Leave the spawned process and return to the RG session with the DCL command LOGOUT.

### 3.12.3 Function keys in MIMER/RG

#### KEYPAD USAGE

<b>HELP</b> <i>PF1</i>	<b>EXIT</b> <i>PF2</i>	<b>PAGE FWD</b> <i>PF3</i>	<b>PAGE BACK</b> <i>PF4</i>
7	8	9	<b>CLEAR</b> /
4	5	6	<b>DEL CHAR</b> ,
1	2	3	<b>SEND/ ENTER</b>
0		<b>INS ↔ OVER</b> .	

#### KEY FUNCTIONS

KEY	FUNCTION
PF1...PF14	Programmed function keys
CLEAR	Clear to the end of field
DEL CHAR	Delete current character
INS/OVER	Set/reset insert mode (toggle)
SEND/ENTER	Send (end of input)
DEL	Delete preceding character
BACKSPACE	Jump to previous field
TAB	Jump to next field
RETURN	Jump to the first field in the next line. If used from the last field in a form, RETURN is interpreted as SEND.

KEY	FUNCTION
CTRL A	Set/reset insert mode (toggle)
CTRL D	Delete character
CTRL E	Jump to end of field
CTRL K	Clear to end of field
CTRL N	Toggle application / numeric keypad
CTRL P	Print screen
CTRL R	Refresh screen
CTRL W	Refresh screen

### 3.12.4 Default file types

#### Input

Sequential files containing RGL report specifications are assumed to have the file type .DAT unless another file type is explicitly entered.

#### Output

Output files generated by the RGPRINT function are given the file type .DAT unless another file type is explicitly entered.

### 3.12.5 The MIMINI initiation file and RG

The initialization file MIMINI may be used to store login procedures for RG. The default name for the file is MIMINI.DAT in the current default device and directory. The initialization file may also be referenced by the logical name MIMINI.

The MIMINI file may also include commands for execution by the RGBATCH function.

### 3.12.6 Linking RG applications

Programs calling RG functions through the RG programming interface must be linked with the object library MIMLIB7:RGLIB as well as the basic functions addressed through the MIMERM7 or MIMERS7 option file. An example follows.

```
$ LINK program,MIMLIB7:RGLIB/LIB,MIMERM7/OPT
```

### 3.13 MIMER/SHD

This module contains the tools needed for developing SH pictures and applications. MIMER/SH is a screen handler for asynchronous terminals only. If programs are to be portable between different terminal types, the FM Forms Manager should be used for screen management.

Version 5.2.2 of MIMER/SHD is used with MIMER version 7.3.

#### 3.13.1 Accessible files

The SHD module includes the following files that are accessible to the user:

MIMEXE7:SHEM.EXE	SH Screen Editor for multi-user mode.
MIMEXE7:SHES.EXE	SH Screen Editor for single-user mode.
MIMLIB7:SHESOU.SHF	Screens used by the SH Editor.

#### 3.13.2 Starting the editor

Please read Section 3.15 in this manual for information on selecting terminal type, compiling pictures, etc.

## 3.14 MIMER/SHR

This module contains the runtime libraries for the SH Screen Handler. MIMER/SH is a screen handler for asynchronous terminals. Applications requiring portability between different terminal types should use the FM Forms Manager for screen management.

The SHR module also contains the SHC screen compiler and the TERMD terminal definition utility.

### 3.14.1 Accessible files in MIMER/SHR

The SHR module includes the following files that are accessible to users:

MIMEXE7:SHC.EXE	Screen Compiler utility.
MIMEXE7:TERMD.EXE	Terminal Definition utility.
MIMLIB7:SHFUNC.DEF	Function key definitions.
MIMLIB7:SHLANG.DAT	Runtime error messages for SHC.
MIMLIB7:SHLIB.OLB	SH runtime library.
MIMLIB7:SHTRMNEW.DAT	Terminal definitions processed by TERMD from TRMNEW.DEF.
MIMLIB7:TRMNEW.DEF	Source code for terminal definitions.

### 3.14.2 Initialization files

The initialization file MIMINI may be used to store login procedures for SH. The default name for the file is MIMINI.DAT in the current default device and directory. The initialization file may also be referenced by the logical name MIMINI.

The terminal type for SH may be set in the initialization file SHINIT.INI in the current device and directory. This file should contain a single line with the terminal number (see Section 3.14.3), preceded by the optional keyword 'TTYTYPE: '. Use of the SHINIT file to determine terminal type is not mandatory (see Section 3.14.4).

### 3.14.3 Terminal types

The following terminal types are supported in the standard distribution:

No	Terminal
--	-----
1	Comex 3380 / Infoton 200
2	VT52 / Teleray 10
3	VT100
203	VT100
4	Tandberg
5	Tandberg 2200 / NOTIS
6	VC404
7	VC414
8	HP 2640 / HP 2622
9	VT220 / VT240 / VT320 / VT340
11	Facit Twist

Other terminal types can be defined by the user with the TERMD utility (see the *MIMER/SH Reference Manual*).

### 3.14.4 Selecting terminal type

If the SH initialization file SHINIT.INI (see Section 3.14.2) is available in the current directory or is addressed by a logical name, the terminal type specified in this file will be automatically used when the SH editor is started.

If there is no SHINIT file, the terminal type set in VMS will be used if possible. The current device type is displayed by the DCL command SHOW TERMINAL, and may be changed with the DCL command SET TERMINAL/Device\_Type.

If the terminal device type is set to 'unknown', a list of available terminal types will be displayed when SH is started, and the user is prompted to select the appropriate type.

Note that any entry in the SHINIT file overrides the VMS terminal type setting.

### 3.14.5 Terminal definitions

Terminal type definitions are stored in source code in the file MIMLIB7:TRMNEW.DEF. The compiled form of this file is found in MIMLIB7:SHTRMNEW.DAT. The language used for defining terminal types is described in the *MIMER/SH Reference Manual*.

Contact your system manager if your terminal does not behave as expected.

### 3.14.6 Terminal-specific key sequences

Below is a list of the default control sequences for VT52, VT100- and VT200-series terminals. The functions are identified by symbolic names as given in the *MIMER/SH Reference Manual*. If you are using another terminal definition, contact your system manager or MIMER representative for a list of terminal-specific key sequences.

A different set of key sequences may have been installed at your site. Contact your system manager if the sequences given here do not appear to work.

---

**Note:** Key sequences are of two types, control and escape. Control sequences (for example CTRL-U) are obtained by pressing CTRL and U **together**. Escape sequences (for example ESC U) are obtained by pressing ESC and U **in sequence**.

---

<i>Name</i>	<i>VT52</i>	<i>VT100</i>	<i>VT200</i>
<b>General control keys</b>			
<Leave>	@	@ or PF1	@ or PF1
<Abort>	!	! or PF2	! or PF2
<Update>	\$	\$ or PF3	\$ or PF3
<EntRef>	&	& or PF4	& or PF4

---

**Control keys for picture definition**

<BegOLine>	CTRL-A	CTRL-A	F12
<EndOLine>	CTRL-E	CTRL-E	F13
<RfrScr>	CTRL-R	CTRL-R	CTRL-R
<InsBLine>	CTRL-L	CTRL-L	F9
<UnDLine>	CTRL-U	CTRL-U	INSERT
<DelELine>	CTRL-K	CTRL-K	REMOVE
<DelCLine>	ESC <CR>	ESC <CR>	F10
<SplLine>	ESC /	ESC /	ESC /
<DelPChar>	DEL	DEL	DEL
<DelCChr>	CTRL-D	CTRL-D	CTRL-D
<UppCase>	ESC U	ESC U	F7
<LowCase>	ESC L	ESC L	F8
<CapCase>	ESC I	ESC C	F11 C
<Over/Ins>	ESC T	ESC T	F14
<EntCom>	ESC K	ESC K	DO
<PFDef>	ESC P F	ESC Q	F17
<PFTag>	ESC .	ESC .	F18
<PFMove>	ESC M	ESC M	F19
<PFNext>	ESC N	ESC N	F20

---

### 3.14.7 The SH Screen Compiler

The screen compiler is started with the command

```
$ RUN MIMEXE7:SHC
```

Note that the compiler operates directly on the picture definition source file. It does not use MIMER/DB, so there is no log-in requirement and no single/multi-user option.

The compiler prompts for the name of the picture definition source file (normally the output file from the editor) and for the names of the two output files. There are no default file names. Default file types are as follows:

Picture definition source file	.SOU
Picture definition object file	.SHF
Coupling source file	.FOR

The operator may define the minimum size of the object file if desired. This size is given in MIMER pages (1 MIMER page = 4 VMS blocks). If no size is specified, object files are created with a minimum size of 10 pages (40 blocks). Sizes less than 3 pages (12 blocks) should not be used, to avoid object file fragmentation.

The coupling file is generated in FORTRAN.

#### Example:

```
$ RUN MIMEXE7:SHC
Picture definition source file: HOTELPIC
Picture definition object file: HOTELPIC
Object file size             : 10
Generated coupling source file: HOTELPIC
```

### 3.14.8 Linking application programs with SH

Application programs using SH must be linked with object code compiled from the coupling file(s) generated by the SH compiler. The picture definition object code is used at runtime and should not be linked with the application. The runtime library for SH routines is linked through the MIMERSSSH or MIMERMSH option file, addressed by the logical names MIMERSSSH7 and MIMERMSH7 respectively.

#### Example

Link an application called HOTAPPL with a coupling file HOTELPIC for multi-user operations (remember that the FORTRAN coupling file must first be compiled with the FORTRAN compiler):

```
$ FOR HOTELPIC
$ LINK HOTAPPL,HOTELPIC,MIMERMSH7/OPT
```

### 3.14.9 Runtime version check facility

A consistency check between the SH software version and input data (picture and terminal definitions) is made at runtime in SH applications.

Inconsistencies between the SH software version and the input files generate the error messages :

```
Version error, definition file  
Version error, terminal def file
```

Picture definition version inconsistencies indicate either that the picture definitions were compiled with an incompatible SHC version or that the application was linked with the wrong version of the runtime library. The error is corrected either by recompiling the picture definitions or relinking the application.

Terminal definition version inconsistencies indicate either that the terminal definitions were generated with the incorrect TERMD version or that the application is linked with the incorrect runtime library. The error is corrected either by regenerating the terminal definitions or relinking the application.

## 3.15 MIMER/UTIL

The UTIL (utilities) module provides several functions required to maintain and create a database: Backup/Restore, Statistics, Readlog, Export/Import and Shadow maintenance.

### 3.15.1 Accessible files

The MIMER/UTIL module includes the following files that are accessible to users:

MIMEXE7:UTILM.EXE                      Utility program - multi-user mode.

MIMEXE7:UTILS.EXE                      Utility program - single-user mode.

---

**Note:** The installation process does not require that all of the above files are generated. For example, the system installer may have chosen to generate only some of these files to save disk space. Users should contact the system administrator if a file they need is missing.

---

### 3.15.2 File handling conventions

#### Export

This utility uses the default file type .EXP for the export file if the file name specified has no type. This file can contain both data and definitions.

#### Import Object Creation

Input files in this phase have the default type .EXP if none is explicitly specified. The default type .IMP is used for the generated import logfiles.

#### Import Data Load

This phase has two input files, with the default types .EXP and .IMP, if no other extension is specified.

### 3.15.3 The MIMINI initialization file and MIMER/UTIL

The initialization file MIMINI may be used to store login procedures for MIMER/UTIL. The default name for the file is MIMINI.DAT in the current device and directory. The initialization file may also be addressed by the logical name MIMINI.



## 4 DATA TYPES USED IN MIMER

### 4.1 Internal DB representation

The MIMER/DB database module uses only two data representations internally: numeric and character. These two types are used to store all data in the database.

The numeric data type stores numbers in packed BCD format with a maximum precision of 45 digits. Every number stored has an exponent with a range of -999 to +999.

The character data type uses the ISO 8859-1 standard character set. This standard specifies graphic characters and the representation of each character. It is a proper extension of the DEC multinational character set with a few exceptions.

The VT300 series terminals support both DEC multinational and ISO 8859-1. Since the two character sets are quite similar it is recommended that any VMS site using VT300 terminals uses the ISO character set.

The lower part (first 128 characters) of ISO 8859-1 is identical to the ordinary ASCII standard. The upper part (last 128 characters) contains 32 control codes followed by the characters shown below:

NS	ı	ç	£	¤	¥		§	¨	©	ª	«	¬	SH	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

## 4.2 External data types supported by MIMER

Any value stored in the database may be read into variables of any of the types listed below. MIMER/DB will perform all necessary conversions and will signal an error if the value to be converted does not fit into the destination type. The data types supported are:

VARCHAR	This data type contains a two-byte length information followed by the number of characters specified.
FIXED CHAR	This data type contains the number of bytes (characters) specified as the length of the variable having this type.
NULL TERMINATED STRING	This data type consists of a string of bytes terminated by a null byte. This type is compatible with C-language strings.
FLOAT	<p><b>VAX</b> Depending on the length specified, the FLOAT data type is implemented as either F_FLOATING (4 bytes), D_FLOATING (8 bytes), or H_FLOATING (16 bytes). ♦</p> <p><b>AXP</b> Depending on the length specified, the FLOAT data type is implemented as either F_FLOATING (4 bytes) or G_FLOATING (8 bytes). ♦</p>
DECIMAL	This data type is a packed BCD number containing a maximum of 18 digits.
SMALL INTEGER	A word consisting of two bytes.
LARGE INTEGER	A longword consisting of four bytes.

Note that MIMER does not support passing data by descriptor.

### AXP

The floating point data types supported on Alpha/VMS are F-float and G-float (as described above). The D-float data type is not supported by MIMER. Please use the /G\_FLOAT switch when compiling programs that should be linked with MIMER libraries.

MIMER does not currently support the use of IEEE floats (S-float or T-float).

H-float is not supported by MIMER on Alpha/VMS since Digital does not support that data type on the Alpha platform.

64-bit integers are not supported by MIMER on Alpha/VMS. ♦

### VAX

G-float is not supported by MIMER on VAX/VMS. ♦

## A USING TAPE DEVICES AS SEQUENTIAL FILES

In exporting database contents to other machine environments, and in the conversion of version 4 databases to version 7, considerable demands are made on sequential file space for large databases. Under VMS, tape devices can be used instead of sequential files, thereby reducing the demand for disk space and generating a physically movable copy of the database in one operation.

Refer to the *Guide to VMS Disk and Magnetic Tape Operations*, available from Digital Equipment, for more information on handling tapes.

### A.1 Writing data to magnetic tapes

#### A.1.1 Preparations

Estimate the number of tapes needed from the size of the database. The write operation will be simplified if all tapes are properly initialized before you start.

Tapes are initialized with the command

```
$ INIT/DENSITY=xxxx device: volume_name
```

During the write operation, requests to change tapes are sent by the Record Management System (RMS) to the operator. In order to receive and respond to these requests, you must log in to another terminal and enable the OPERATOR privilege:

```
$ SET PROC/PRIV=OPER  
$ REPLY/ENABLE=TAPE ! to enable tape messages
```

#### A.1.2 Writing to the tape unit

Load the tape station with an initialized tape and mount the tape volume:

```
$ MOUNT tape_device
```

Now start the program that creates the sequential file (e.g. Export/Import or CONV47). When the program asks for the name of the sequential file, give a file name on the tape device:

```
Sequential file name: tape_device:filename
```

When the tape is full, a message will be displayed on the operator terminal(s) where tape messages are enabled. Remove the full tape from the station and load a new, initialized tape. **Do not dismount the tape.** Resume the write operation on the new tape by giving a REPLY from the operator terminal:

```
$ REPLY/TO=nnn
```

where 'nnn' is the number of the operator request, as given in the message displayed on the operator terminal.

When the write operation is complete, dismount the tape volume and remove the last tape from the station:

```
$ DISMOUNT tape_device
```

## A.2 Reading sequential data from a tape

Load the first tape containing the sequential data and mount the tape volume:

```
$ MOUNT tape_device
```

Start the application that will read the sequential data (e.g. Export/Import). When the program asks for the name of the sequential file, give the appropriate file name on the tape device:

```
Sequential file name: tape_device:filename;1
```

Note that you should give the version number of the file explicitly. If you do not give a version number, RMS will read through **all** the tapes in sequence, to find the highest version number, and then read the data in a second pass through the tapes.

Requests to mount continuation tapes will be sent to the operator terminal as necessary.

## **B EXIT ROUTINES**

This appendix considers the use of the EXITC1 error handling routine in application programs written with the MIMER Level 2 programming interface and run in a version 7 environment. Such programs are typically those originally developed in a version 3 or version 4 system. The use of the Level 2 interface for application programs developed under version 7 is discouraged.

### **B.1 General considerations**

In version 4 of MIMER, errors at the lowest user-programming level of the database manager (Level 2) were trapped by the EXITC1 exit routine, documented in the *MIMER/DB version 4 Reference Manual*. User-written EXITC1 routines could be linked with application programs for special error-handling purposes.

In MIMER version 7, exit routines are reduced to a minimum, and low-level errors are handled by internal routines. The routine SETER2 may be used to control whether a program will abort or continue when an error occurs. There is no general exit routine that the user may modify to control the individual error consequences. Any specific error handling that is required for an application program must be written in the form of explicit tests of the error code returned from the appropriate routine.

The reason for eliminating user-written exit routines from version 7 of MIMER is that the standard DB runtime library is implemented as a shared library. To maintain compatibility with application programs written for version 4 that depend on user-written EXITC1 routines, support for EXITC1 can be forced in version 7 by linking in non-shared mode.

## B.2 Using standard error handling

The SETER2 routine allows the user to set error handling to one of four standard modes, covering program abort or continue, return error code and display error message (see the *MIMER Runtime Libraries Programmers Manual, Low Level Interface* for details).

In general, it is recommended that application programs written for version 4 are modified to use SETER2 to control error handling wherever possible. If a program is dependent on a user-written EXITC1 routine, this may not be practicable. In that case, the application may be linked in non-shared mode in version 5, giving access to the user-written EXITC1 routine.

## B.3 Advantages of standard error handling

Using standard error handling (SETER2) gives a number of advantages:

- future support for the application program is assured (Sysdeco Mimer AB does not guarantee indefinite support for EXITC1 in future versions of MIMER)
- improved portability between machine types (support for EXITC1 may not be available on all machines in version 7).