

MIMER

User's Guide for UNIX[®]

Version 7.3

Copyright

©

1996

Sysdeco

Mimer

AB

MIMER version 7.3 User's Guide for UNIX

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

| | | |
|----------|---|------|
| 1 | INTRODUCTION | |
| 1.1 | Documentation objectives | 1-1 |
| 1.2 | Organization of the documentation | 1-1 |
| 1.3 | Conventions | 1-2 |
| 1.4 | Important terms | 1-3 |
| 1.4.1 | MIMER structure terms | 1-3 |
| 1.4.2 | MIMER administrative terms | 1-4 |
| 1.5 | Acronyms and trademarks | 1-4 |
| 2 | GENERAL INFORMATION | |
| 2.1 | Structure of a MIMER product installation | 2-1 |
| 2.2 | MIMER user related environment symbols | 2-1 |
| 2.2.1 | Defining environment variables | 2-1 |
| 2.2.2 | Optional environment variables | 2-2 |
| 2.2.3 | Shared library lookup | 2-3 |
| 2.3 | MIMER database configuration | 2-3 |
| 2.3.1 | Single-user vs. multi-user database | 2-3 |
| 2.3.2 | Multiple multi-user database systems | 2-4 |
| 2.4 | Starting MIMER programs | 2-4 |
| 2.4.1 | Finding the executable program files | 2-4 |
| 2.4.2 | Logging in to MIMER | 2-5 |
| 2.4.3 | The MIMINI file | 2-5 |
| 2.5 | MIMER databank file handling | 2-5 |
| 2.5.1 | General features | 2-5 |
| 2.5.2 | Databank files in single-user database | 2-6 |
| 2.5.3 | Databank files in multi-user database | 2-6 |
| 2.5.4 | Conversion between single- and multi-user databases | 2-6 |
| 2.6 | File I/O in MIMER | 2-7 |
| 2.6.1 | MIMER direct access I/O | 2-7 |
| 2.6.2 | MIMER sequential I/O - general | 2-7 |
| 2.6.3 | Sequential files - extended filename syntax | 2-8 |
| 2.7 | Error codes | 2-8 |
| 2.7.1 | MRS errors | 2-8 |
| 2.7.2 | MIMER/DB system access errors | 2-9 |
| 2.7.3 | Direct access I/O errors | 2-11 |
| 2.7.4 | Sequential access I/O errors | 2-12 |
| 2.8 | Linking C-applications | 2-13 |
| 2.8.1 | Optimizing compilers | 2-13 |
| 2.9 | Executing UNIX commands from MIMER modules | 2-13 |
| 2.10 | MIMER command line editing facility | 2-14 |
| 2.11 | On-line documentation | 2-14 |
| 3 | MODULE SPECIFIC INFORMATION | |
| 3.1 | MIMER/DB | 3-1 |
| 3.1.1 | Accessible files in MIMER/DB | 3-1 |
| 3.1.2 | Using singleinstall | 3-3 |

| | | |
|--------|--|------|
| 3.1.3 | Exit routines | 3-3 |
| 3.2 | MIMER/ESQL | 3-3 |
| 3.2.1 | Accessible files in MIMER/ESQL..... | 3-3 |
| 3.2.2 | UNIX compilers supported | 3-4 |
| 3.2.3 | ESQL command syntax..... | 3-4 |
| 3.2.4 | Compiling the example.ec program..... | 3-4 |
| 3.2.5 | Compiling the dsqsamp program..... | 3-5 |
| 3.2.6 | Compiling the blobsamp program | 3-5 |
| 3.2.7 | File handling conventions..... | 3-6 |
| 3.2.8 | Type definitions | 3-6 |
| 3.3 | MIMER/FMD | 3-6 |
| 3.3.1 | Accessible files in MIMER/FMD | 3-6 |
| 3.3.2 | File handling conventions..... | 3-7 |
| 3.3.3 | FMD keypad usage..... | 3-7 |
| 3.4 | MIMER/FMR..... | 3-7 |
| 3.4.1 | Accessible files in MIMER/FMR | 3-7 |
| 3.4.2 | Supported terminal types..... | 3-7 |
| 3.4.3 | Keypad mode | 3-8 |
| 3.4.4 | Linking applications to MIMER/FMR..... | 3-8 |
| 3.5 | MIMER/ISQL | 3-9 |
| 3.5.1 | Accessible files in MIMER/ISQL..... | 3-9 |
| 3.5.2 | Keyboard interrupt | 3-9 |
| 3.5.3 | ISQL keypad usage..... | 3-9 |
| 3.5.4 | The MIMINI initiation file | 3-10 |
| 3.6 | MIMER/PGD..... | 3-10 |
| 3.6.1 | Accessible files in MIMER/PGD..... | 3-10 |
| 3.6.2 | Functionality of the code generator | 3-11 |
| 3.6.3 | File handling conventions..... | 3-11 |
| 3.6.4 | The pgrun command script | 3-11 |
| 3.6.5 | Keyboard interrupt | 3-12 |
| 3.6.6 | The MIMINI and PG initiation files | 3-12 |
| 3.7 | MIMER/PGR | 3-13 |
| 3.7.1 | Accessible files in MIMER/PGR..... | 3-13 |
| 3.7.2 | Linking a PG application..... | 3-13 |
| 3.8 | MIMER/PI | 3-14 |
| 3.8.1 | Accessible files in MIMER/PI..... | 3-14 |
| 3.8.2 | Linking an application that uses MIMER/PI..... | 3-14 |
| 3.8.3 | Exit routines | 3-14 |
| 3.9 | MIMER/QF..... | 3-15 |
| 3.9.1 | Accessible files in MIMER/QF | 3-15 |
| 3.9.2 | QF keypad usage | 3-15 |
| 3.10 | MIMER/QL..... | 3-16 |
| 3.10.1 | Accessible files in MIMER/QL | 3-16 |
| 3.10.2 | Keyboard interrupt | 3-16 |
| 3.10.3 | QL keypad usage | 3-16 |
| 3.10.4 | The MIMINI initiation file and QL..... | 3-17 |
| 3.11 | MIMER/RG | 3-17 |
| 3.11.1 | Accessible files in MIMER/RG | 3-17 |
| 3.11.2 | RG keypad usage..... | 3-18 |
| 3.11.3 | The MIMINI initiation file and RG | 3-18 |
| 3.11.4 | Linking RG applications..... | 3-18 |
| 3.12 | MIMER/SHD..... | 3-19 |
| 3.12.1 | Accessible files in MIMER/SHD..... | 3-19 |
| 3.12.2 | Terminal specific key sequences..... | 3-19 |
| 3.13 | MIMER/SHR | 3-22 |
| 3.13.1 | Accessible files in MIMER/SHR..... | 3-22 |
| 3.13.2 | Initiation files..... | 3-22 |
| 3.13.3 | Terminal specific key sequences..... | 3-22 |
| 3.13.4 | The SH compiler..... | 3-23 |

| | | |
|----------|---|------|
| 3.13.5 | Linking application programs with SH..... | 3-24 |
| 3.13.6 | Runtime version check facility | 3-24 |
| 3.14 | MIMER/UTIL | 3-25 |
| 3.14.1 | Accessible files in MIMER/UTIL | 3-25 |
| 3.14.2 | File handling conventions..... | 3-25 |
| 3.14.3 | Using mimchop..... | 3-26 |
| 3.14.4 | Using mimunchop | 3-26 |
| 3.14.5 | The MIMINI initiation file | 3-26 |
| 4 | DATA TYPES USED IN MIMER/DB | |
| 4.1 | Internal DB representation | 4-1 |
| 4.2 | External data types supported by MIMER..... | 4-1 |
| 5 | INSTALLING A SINGLE-USER DATABASE SYSTEM | |
| 5.1 | The installation procedure | 5-1 |
| 5.2 | Installation errors..... | 5-4 |
| A | EXIT ROUTINES | |
| A.1 | General considerations | A-1 |
| A.2 | Standard error handling..... | A-1 |
| A.2.1 | Using standard error handling | A-1 |
| A.2.2 | Advantages of standard error handling..... | A-2 |
| B | TERMINAL DEFINITIONS | |
| B.1 | Modules concerned..... | B-1 |
| B.2 | Terminfo files | B-1 |
| B.3 | Common key definitions..... | B-2 |
| B.4 | Function key layouts..... | B-3 |

1 INTRODUCTION

1.1 Documentation objectives

This guide describes the UNIX specific features of all MIMER modules, such as file specifications, etc. The guide is intended for the application developer. It is assumed that the reader is familiar with MIMER and with the UNIX operating system.

This guide does not describe how to use MIMER in general, and should be used as a supplement to the reference manuals for the separate MIMER modules.

Late-breaking information, keypad diagrams and some installation-specific details may be found in the *MIMER Release Notes for UNIX*.

1.2 Organization of the documentation

The documentation describing installation and maintenance of MIMER on a UNIX system is divided into two guides:

MIMER Installation Guide for UNIX:

This guide is intended to help the UNIX system administrator prepare the UNIX system and install MIMER.

MIMER Administration Guide for UNIX:

This guide supports the MIMER system administrator in configuring and maintaining the MIMER software and database systems.

The following document is a guide for users of the MIMER system:

MIMER User's Guide for UNIX:

This guide is a supplement to the general MIMER reference manuals, and contains UNIX specific information. This guide is intended for users of a MIMER system under UNIX.

The following document contains machine dependent information:

MIMER Release Notes for UNIX:

This document includes comments on a machine specific implementation of MIMER, such as keypad layouts, and commands specific to that machine.

1.3 Conventions

The following terms and conventions are used in this guide:

| | |
|----------------------|--|
| ~username | Following the normal c shell syntax, the tilde (~) represents the name of the UNIX home directory for the given username. |
| environment variable | A shell variable that can be assigned a string value. See your UNIX documentation for a more thorough discussion. |
| group id | The numerical identification codes used by UNIX to identify a group of users when checking access privileges to files or other UNIX resources. |
| user id (uid) | The numerical identification codes used to identify a specific login name. The identification number is assigned when the account is created on the particular UNIX system and is used by UNIX when checking access privileges to files or other UNIX resources. |
| pathname | A series of directory names separated by “/” characters and ending in a directory or filename. |
| absolute path name | A path name beginning with the “/” character, which means “from the root directory”, when used as the first character. |
| search path | A series of pathnames which are tried one after the other when looking for a specific file. |
| superuser | The user who has special privileges, such as no file or access limitations. This user is ordinarily responsible for performing UNIX administration tasks and has a login name called root. |

Note! Long pathnames 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 UNIX.

1.4 Important terms

The following terms are important in the discussion presented in this document.

1.4.1 MIMER structure terms

| | |
|----------|--|
| database | A database is a collection of tables, databanks, idents, domains, etc. All of these objects are defined in the data dictionary, SYSDB. A UNIX system may contain several MIMER databases operating simultaneously, each with its own SYSDB, but no information may be shared between different MIMER databases. Each database has a unique name registered together with the location of its SYSDB databank in the file <code>/etc/sqlhosts</code> . |
| databank | A databank corresponds to one UNIX file or raw device. A databank may contain several tables. |
| table | Tables (or relations) hold all information in the relational database. A table may not be split over several databanks, but a databank can hold several tables. |
| shadow | 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. |

For other terms such as ident and domain, see the *MIMER System Management Handbook*.

1.4.2 MIMER administrative terms

- MIMERADM** **The MIMER software administrator.** This is a specific UNIX user, created to own and administer the libraries, executable files, data files, etc. distributed with MIMER. Only one user is to be created for this purpose, and, for example, could be given the UNIX user name **mimer7**, which will be used in examples shown in this document.
- ~MIMERADM** The HOME directory of the MIMERADM user. The installation path **/opt/products/mimer7** will be used in examples shown in this document.
- MULTIADM** **The MIMER multi-user database administrator.** This is a specific UNIX user, created to own and administer a specific database. One user should be created for each MIMER multi-user system to be installed. Suitable UNIX user names are **multi1**, **multi2** etc. Note that the user name given the MULTIADM user will become the main database name, although aliases may be created for each database. Also note that this user must not be the MIMERADM user. The mentioned names will be used in examples shown in this document.
- ~MULTIADM** The HOME directory of the MULTIADM user. The database paths **/d1/multi1** and **/d2/multi2** will be used in examples shown in this document.

1.5 Acronyms and trademarks

- MRS** MIMER Release and Security.
- ESQL** Embedded SQL
- UNIX** UNIX is a trademark registered by X/Open Company Ltd.

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

2 GENERAL INFORMATION

2.1 Structure of a MIMER product installation

The structure and contents of an installed MIMER product is described in detail in the *MIMER Administration Guide for UNIX*.

The most important directories to know about for MIMER users in general are the following:

| | |
|----------------------------|--|
| <code>~MIMERADM/bin</code> | all executable MIMER programs reside here. |
| <code>~MIMERADM/lib</code> | all MIMER libraries (shared libraries and archive files), used when linking an application, reside here. |

2.2 MIMER user related environment symbols

In general, nothing special needs to be done to enable a UNIX user to use either the MIMER modules (in multi-user mode) or a MIMER based application. However, to ensure proper operation of the MIMER system, certain UNIX environment variables may need to be defined.

2.2.1 Defining environment variables

Remember that programs started by a user will have complete access to these variables (via the process environment) only if they are defined globally system wide or if they are marked for automatic export to the environment of subsequently executed commands. The way of marking environment variables is dependent upon which shell the user is running, see the following examples:

Bourne shell:

```
MIMER_HOME=/opt/products/mimer7; export MIMER_HOME
```

C-shell:

```
setenv MIMER_HOME /opt/products/mimer7
```

Korn shell:

```
export MIMER_HOME=/opt/products/mimer7
```

2.2.2 Optional environment variables

As an option, one or more of the following environment variables can be set by a user to alter the behaviour of the MIMER system:

MIMER_DATABASE

Used to select a MIMER multi- or single-user database to connect to. Defines a database name that will be looked up in the `/etc/sqlhosts` file, where connection information for all databases in a network are defined. For a local database, a databank search path is specified and for a remote database the network protocol information is given. If the `MIMER_DATABASE` environment variable is undefined, the `/etc/sqlhosts` file contains a `DEFAULT` entry, pointing to a `LOCAL` or `REMOTE` database, which will be used.

MIMER_HOME

Should point out the MIMER installation to use, i.e. the `~MIMERADM` directory. MIMER uses the path defined in the variable to find specific files in run-time.

MIMER_KEYFILE

Points to the file where the encrypted MRS license resides. All MIMER programs need access to this file in order to execute. If the environment variable is undefined or points to an unreadable file, the file `/mimkey7` is used.

MIMER73_PATH

Used as a search path for sequential files and other data files used by the MIMER system. It is relevant when opening user specified sequential files such as MIMER/SH picture source, system import/export files, etc.

To set up several directory pathnames in a path list, separate the names with colons (i.e. "pathname1:pathname2"). Each directory in the search path of the variable is checked, one after the other, in an attempt to locate the desired file. If you have no `MIMER73_PATH` environment variable defined, MIMER will attempt to access files relative to the current working directory.

New files are created in the first writable directory mentioned in `MIMER73_PATH`, but only after all of the succeeding directories have been checked for the existence of a file with the same name. If a filename is found in a directory mentioned in search path of the variable, an error is returned to avoid accidental or deliberate "aliasing" of filenames.

MIMER73_NICE

Used when the priority of the MIMER/PG C-generator process is to be changed. The default value used for the UNIX `nice` command is 10. If the `MIMER73_NICE` environment variable is set, the given value is used instead. See Chapter 3.6, MIMER/PGD.

Apart from the MIMER specific environment variables mentioned, there are several ordinary UNIX shell variables that need modification:

PATH

This is not used by MIMER modules, but as usual, it must be defined so that MIMER and user application programs and command scripts can be found by UNIX command shells. Be sure your PATH refers to only one MIMER software release or version to avoid executing an unexpected version of a program.

TERM

If you are using a MIMER module or application program which uses MIMER/FM or MIMER/SH, this variable must be set to one of the terminal types supported by MIMER, and listed in the *MIMER Release Notes for UNIX*.

2.2.3 Shared library lookup

Another environment variable that may be needed in the environment of each MIMER user is the variable used by the load editor (ld) to locate shared libraries when executing programs. The name of this variable is operating system dependent and for your specific environment it can be read out from the `~MIMERADM/dat/makeopt` file, as the string value assigned to the `MIM_SHLIB_ENV` symbol.

The definition made for this environment variable should include the `~MIMERADM/lib` directory, and in most cases also the `/usr/lib` and `/lib` directories (for UNIX system shared libraries) as shown in the following example:

```
LD_LIBRARY_PATH="/opt/products/mimer7/lib:/usr/lib:/lib"
```

If using dynamically linked programs the corresponding environment variable most certainly needs to be defined properly.

2.3 MIMER database configuration

2.3.1 Single-user vs. multi-user database

On a computer, one or more MIMER single-user database systems, one or more MIMER multi-user database systems, or a combination of both types in parallel, can be installed. A single-user database system permits only one user at a time to access a database, although many users may be authorized to use the database at different times. A multi-user database system permits a pre-defined number of users to access the same database simultaneously. (Practical limits depend on the bufferpool size, the amount of UNIX kernel resources provided for MIMER during installation, and the MRS-license. The maximum number of users per multi-user database system is also set when the system is started.)

The databanks, application program source and object code are identical between single- and multi-user systems. The only thing that determines whether applications are single- or multi-user is the MIMER/DB library with which the application is linked. An end-user may not be aware of whether the program runs in single- or multi-user mode.

Most (but not all) MIMER executable programs access MIMER/DB and are available in both single- and multi-user versions. The last letter of a program's executable filename indicates which type it is (**m** for multi-user or **s** for single-user). Thus the program **isqlm** runs ISQL in multi-user mode, and **utils** is the single-user version of the UTIL program.

2.3.2 Multiple multi-user database systems

Several separate MIMER multi-user database systems can be run in parallel on the same UNIX system. Guidelines for installing several MIMER multi-user systems are described in the *MIMER Administration Guide for UNIX*. There is no fixed limit on the number of MIMER multi-user database systems supported on one machine. Practical limits are determined by the manner in which your UNIX operating system has been installed and configured, and the number of MIMER multi-user system users that the machine is licensed for (see the *MIMER Installation Guide for UNIX* and the *MIMER Administration Guide for UNIX* for more details). Each multi-user database system is owned and controlled by the associated MULTIADM.

Each MIMER system must have completely separate sets of databanks. No databanks are shareable between separate MIMER databases.

An application may be connected to several databases simultaneously, although only one connection is active at one time. The SQL CONNECT, DISCONNECT and SET CONNECTION statements are used to manage database connections (see the *MIMER/SQL Interactive Users Manual* or *MIMER/SQL Reference Manual*).

2.4 Starting MIMER programs

2.4.1 Finding the executable program files

All MIMER programs are started by giving the relevant name on the command line (except for the ESQL module). All executable MIMER modules are found in the `~MIMERADM/bin` directory.

Detailed information and the names of all executable program files available within each module are given in chapter 3 under the respective section for each module.

To enable your command shell to find the MIMER executable files, the path of the directory with the MIMER executable files must be included in your PATH definition. The PATH definition is used by your command shell to locate files that are to be executed.

2.4.2 Logging in to MIMER

MIMER has a security system that requires each user to identify themselves before the database management system can be used. Most MIMER modules have a login screen, which displays the MIMER logo and "Username" and "Password" prompts. The user is allowed three attempts to log in before the program terminates. Use the MIMER username and password assigned to you by the MIMER system administrator, not your UNIX username/password, to log in to MIMER applications.

If your current operating system username is defined as an OS_USER ident in MIMER, you can log in to MIMER simply by pressing the enter key at the "Username" prompt.

2.4.3 The MIMINI file

All MIMER modules that display the login screen described above also accept login information from the MIMINI file. The default name for the file is **MIMINI** in the current working directory. In addition to the MIMER Username and Password which can be put in this file, some modules accept additional information. See the reference manual for a specific module to get detailed information. For an example of the contents of the MIMINI file, see the *MIMER/QL Reference Manual*. (See also detailed module information later on in this guide).

2.5 MIMER databank file handling

2.5.1 General features

Databank files are always accessed by **direct access I/O** routines (see section 2.5.1) in the database manager.

Filenames for all databanks except SYSDB are stored in the data dictionary in the exact form that they were entered when the databanks were created. Users are strongly advised not to use absolute file pathname specifications (i.e. beginning with "/") when creating databanks. Use of relative pathnames makes databank relocation much easier.

When an application is started, the file containing the SYSDB databank (data dictionary) must be found in the first directory of the search path given in **/etc/sqlhosts**.

All other databank filenames are stored in the data dictionary. If absolute pathname specifications were given when the databanks were created, the locations are unambiguous. If a relative pathname is given, i.e. not beginning with "/", the search path from **/etc/sqlhosts** is used to find the file.

The filename specified for a databank can be displayed from the data dictionary with the DESCRIBE command available in ISQL or BSQL (see the *MIMER/SQL Interactive Users Manual*).

2.5.2 Databank files in single-user database

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

For single-user databases relative directories may be part of the search path in */etc/sqlhosts*.

For installation of a single-user database system see chapter 5.

Usually the database called SINGLE, that refers the current directory in the example version of the *sqlhosts* file, is used when accessing a single-user database.

2.5.3 Databank files in multi-user database

All databank file handling in MIMER multi-user database systems is performed by the MIMER I/O server processes, leading to that databank files in multi-user database systems are created by the MULTIADM user.

The location of SYSDB for a given system is defined in the */etc/sqlhosts* file, i.e. the primary path for the defined database. All other databank locations for the system are identified through the data dictionary in the SYSDB databank, perhaps in combination with the definition of the database in the */etc/sqlhosts* file (see the *MIMER Administration Guide for UNIX* for details in this matter).

Mimer users who create databanks in multi-user database systems do not own the corresponding files in the UNIX context, and these files will not be placed in the user's directory structure unless this is deliberately specified (which, for security reasons, is not recommended).

2.5.4 Conversion between single- and multi-user databases

If a database (SYSDB and all the associated databanks) created in single-user mode is to be used in MIMER multi-user systems or vice versa, certain precautions must be observed with regard to the definition of databank files:

- File names must be correctly specified in both modes. If no absolute paths are stored as databank file name in the data dictionary, only the */etc/sqlhosts* need to be updated to point out the directories in question. If absolute path names are used the files must remain in current locations or the SQL 'ALTER DATABANK' statement must be used to update the file names atored in the data dictionary. See the *MIMER Administration Guide for UNIX* for details in this matter.
- Files created in single-user mode must be accessible for read and write operations by the actual MULTIADM user if they are to be used in multi-user mode. This is because file access is performed by the MIMER I/O process (which runs with the MULTIADM's user id). The creator of the file should change the UNIX file ownership of the databank to the MULTIADM's user id.

- Conversely, files created in multi-user mode must be accessible for read and write by the individual UNIX users who need to access the databanks in single-user mode. Since multi-user databank files are created by the MIMER I/O processes, protection on these files may only be changed by the MULTIADM user or by the superuser (root). If a multi-user system has opened a databank, no single-user access to that databank is permitted.

2.6 File I/O in MIMER

Two types of file I/O are performed by the MIMER product, Direct/Block oriented (direct access) and Sequential.

2.6.1 MIMER direct access I/O

The direct access I/O routines are used internally by MIMER to access databank files and certain other internal files. Their use is not supported for user applications.

MIMER databank files are always created with the attribute value 2600 (octal). This specifies;

- read and write access to the user id of the process creating the databank,
- no access by other user id's or members of the current effective group and,
- that file locking operations requested by the MIMER direct access I/O routines are to be enforced by the UNIX kernel.

In addition to normal UNIX files, raw device files are supported. Their use can result in dramatically improved response times for large and/or heavily accessed databanks. The use of raw device files for MIMER databanks is further described in the *MIMER Installation Guide for UNIX*.

2.6.2 MIMER sequential I/O - general

MIMER's sequential I/O routines function in a manner largely equivalent to the UNIX standard I/O package when used with the **gets()** and **puts()** functions. MIMER sequential I/O always involves either reading or writing from the beginning to the end of a file in a sequence of variable- or fixed-length records delimited by newline characters in the ordinary UNIX fashion for such cases.

Unlike some operating systems, it is not a good idea to use the same filename as both input and output files. If the same filename is used, it could result in the silent destruction of the input file and/or an error message indicating that the file is "locked by another user".

2.6.3 Sequential files - extended filename syntax

Normally, the name of a file which is to be opened by the MIMER sequential I/O routines should follow the same rules as a filename in any other context on the given UNIX system. As discussed earlier in this chapter, the environment variable, MIMER73_PATH, can be used to “fill out” the pathnames which are supplied.

2.7 Error codes

2.7.1 MRS errors

The MRS system is activated every time a MIMER module is started. This system checks to see if the current module is included in the specific license. If not, you will not be allowed to start the module. It also checks a number of other items such as machine type and date and compares this to an encoded software key to see if the installation is licensed to use the module.

The software key is usually located in the file **/mimkey7** (in the root directory). However, the system administrator may have specified an alternate filename for the MRS key. In this case, you must define the MIMER_KEYFILE environment variable to point to the MRS file. Every MIMER user must have read access to the key file. If the MRS system finds any errors or mismatches, an error message is displayed and the program is terminated. See the *MIMER Administration Guide for UNIX* for a further description of the MRS system.

When an error is detected the line “MIMER-MRS Error” is displayed along with one of the following error messages:

| Error message | Description |
|------------------------|--|
| Incorrect software key | Syntax error - the software key was incorrectly entered. |
| Time limit exceeded | The “last date of use” has expired for the module. |
| Illegal CPU id | The software is not licensed to run on this machine. Please contact your MIMER agent. |
| Illegal module id | The installation is not authorised to use the module. |
| Illegal version | The installation is not licensed to use this version of the module/function. |
| Unauthorized use | The installation is not authorised to use the module/function. |
| File not found | The MRS file is absent or unreadable. Check permissions and the value of the MIMER_KEYFILE environment variable. |

Other error messages that may appear are described later in this section.

2.7.2 MIMER/DB system access errors

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

Note: An error described as "fatal" will terminate execution of the current program.

Single-user system

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 fatal error -16144
        Cannot open databank SYSDB
        file sysdb7.dbf locked by other user
```

A common start-up error in single-user systems is that the location of the SYSDB databank file is incorrectly defined (e.g. the location defined by the environment variable MIMER_DATABASE in combination with the path specified in */etc/sqlhosts* is invalid). This is signalled by the error message:

```
MIMER/DB fatal error -16142
        Cannot open databank SYSDB
        file sysdb7.dbf not found
```

Multi-user system - user errors

The following errors indicate that the user is operating in a way which is not compatible with the multi-user system. The user violates some state or limitation set for the multi-user database system by the system's MULTIADM.

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

This error occurs when a user tries to connect to a multi-user system that has not been started.

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

This error occurs if a user tries to connect to a multi-user system which is currently closed for user logins. Normally this indicates that the system is scheduled to shutdown within a short time.

```
MIMER/DB fatal error -18903
        Access denied to MIMER multi-user system
```

This error occurs if a user does not have search permission to all directories in the path of the MIMER multi-user system's home directory. The system can be setup for **owner** access, **group** access or **world** access. If **group** access is employed, the user must belong to the group having search permission in all directories in the MULTIADM's path to be allowed access to the multi-user system or else the multi-user system must be opened for **world** access.

Multi-user system - system errors

The following errors are caused by internal problems in the MIMER multi-user system and not directly by the user. Normally the error message is accompanied by a more detailed description of the error, sent to the user and the MULTIADM as UNIX mail.

```
MIMER/DB fatal error -18904
    Unable to attach to multi-user system,
    no response
```

This error occurs if a user could not be attached to the multi-user system because there are problems obtaining the appropriate service from the operating system.

```
MIMER/DB fatal error -18920
    Machine dependent error -18920, please
    refer to users guide for explanation
```

The user request failed because execution was interrupted by an unexpected signal.

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

The user's database request failed because the system could not allocate enough memory.

```
MIMER/DB fatal error -18922
    Machine dependent error -18922, please
    refer to users guide for explanation
```

The user request failed because there were too many messages in the message queues of the system.

```
MIMER/DB fatal error -18923
    Machine dependent error -18923, please
    refer to users guide for explanation
```

This error occurs if a user logs on to a multi-user system which has already reached the maximum number of logged on users. The user has to wait until somebody logs off the multi-user system.

```
MIMER/DB fatal error -18924
    Machine dependent error -18924, please
    refer to users guide for explanation
```

The user request failed because the system was found passing a bad parameter to a system call.

```
MIMER/DB fatal error -18925
    Machine dependent error -18925, please
    refer to users guide for explanation
```

This error is reported if a fatal truncation of data occurred when the user's request was executed.

MIMER/DB fatal error -18926
 Machine dependent error -18926, please
 refer to users guide for explanation

If this error occurs the request has been prematurely returned from the multi-user system servers.

MIMER/DB fatal error -18901
 Multi-user system not started

An attempt was made to run a multi-user application using a multi-user system that has not been started.

MIMER/DB fatal error -18903
 Access denied to MIMER multi-user system

An attempt was made to access a multi-user system to which access is denied.

2.7.3 Direct access I/O errors

The following error codes may be returned from direct access I/O operations on databank files. The solution to the error situation is, in most cases, UNIX-specific, and the user should refer to the relevant UNIX documentation when necessary.

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 filename

Invalid character or space specified in the physical databank name.

2 File not found

No databank file was found with the specified name. Check the location of the system databanks if this error code is returned in response to a MIMER login attempt (particularly in single-user systems), or check the physical filename stored in the data dictionary for other databanks. Also check to see that the definition in `/etc/sqlhosts` is valid for the database given in the `MIMER_DATABASE` environment variable.

3 File not accessible

The user is not allowed to create or access this databank due to UNIX file protection attributes. If this is an attempt to create a databank, check that the definition in `/etc/sqlhosts` is valid for the database given in the `MIMER_DATABASE` environment variable.

4 File locked by other user

This databank file is currently open by another process, and is not available for other access.

5 Too many opened databanks

There is a limit on the number of concurrently open databanks. The exact number is both MIMER and UNIX configuration dependent. Either remove some databanks, or reconfigure UNIX and/or MIMER to allow more open databank files.

- 6 **File create error**
File could not be created or extended - partition full.
- 7 **File create error**
File could not be created - disk quota exceeded. This error will not occur in most UNIX implementations.
- 8 **Not used**
- 9 **Other error**
Other error during databank open.

In addition, while executing an application, you may get a message reporting that a databank cannot be extended. This can occur because the file system is full, in which case the databank must be moved (when not open by MIMER) to another file system. Alternatively additional file space must be made available in the current file system. If the file is larger than the UNIX **ulimit** on the size of a single file, the UNIX superuser must use the MIMER **dbextend** utility to increase the size of the databank in question. This can not be done while the databank is in use, so any multi-user system using the databank must be stopped. After the databank size has been increased, or more room has been made available on the file system in question, the operation can be retried. See the *MIMER Installation Guide for UNIX* for details.

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 filename**
Invalid character specified in the filename.
- 2 **File not found**
No physical file was found with the specified name. Check the MIMER73_PATH environment variable.
- 3 **File protection violation**
- 4 **File locked by other process**
These errors occur if the file is locked by another process or if the same process attempts to use a given file for both the input and output.
- 5 **Too many opened files**
There is a limit on the number of concurrently open files. The exact number is both MIMER and UNIX configuration dependent. Close any files you are not using and it should be possible to open a new one. If this does not work consult your MIMER or UNIX administrator.
- 6 **File create error**
File could not be created or extended - partition full.
- 7 **File create error**
File could not be created - disk quota exceeded. This should not occur in most UNIX implementations.
- 8 **Not used**
- 9 **Other error**

In addition, the same situation may occur with sequential I/O, as with direct access I/O, in that a file system may grow to the UNIX **ulimit**. It is, however, possible for a superuser to increase the permissible size of a sequential output file. This is a limitation imposed by the specific UNIX kernel, not by MIMER. If the application allows it, you may be able to close the file in question and continue output to another file. The UNIX superuser can concatenate the two files later if your application cannot read them consecutively.

2.8 Linking C-applications

An example makefile for linking different kinds of applications can be found in `~MIMERADM/dat/ex_makefile`. This example makefile can be copied to the local source code directory where it can be modified to fit your application. Some symbols in the file must be edited to fit your application or installation. These are documented by comments in the makefile.

2.8.1 Optimizing compilers

In general, some application programs may give incorrect results under certain circumstances if the object code is optimized by the compiler. The problem originates primarily from the fact that an optimizing compiler may not always recognize the significance of binding a variable to a column name in embedded SQL or ODBC.

To avoid this problem, all applications can be compiled without the `-O` option. If compiler optimization is required, considerable care should be taken. If you have an ANSI standard compiler, you can declare variables used in this manner as **volatile**. Some other compilers support “pragmas” which allow you to turn optimization off for specific code segments.

2.9 Executing UNIX commands from MIMER modules

Many MIMER modules have a command named `COMMAND`. This command creates a child process to execute the UNIX command given as the parameter to `COMMAND`. If no parameter is given, the MIMER module will “sleep” while the user interacts with a newly created copy of the user’s default command shell (as determined by the `SHELL` environment variable). If the `SHELL` environment variable is not valid, the `COMMAND` command will return an error message. To terminate the process and return to the interactive session, issue the UNIX command **exit** or press the end-of-file key (usually **ctrl-d**).

2.10 MIMER command line editing facility

Emacs and Korn Shell styled command line editing is available in those MIMER programs that use a line-oriented interface, such as in BSQL. The following functions are available:

| | |
|--------|---|
| ctrl-a | Move to beginning of command |
| ctrl-b | Move backwards in command |
| ctrl-d | Delete current character |
| ctrl-e | Move to end of command |
| ctrl-f | Move forwards in command |
| ctrl-h | Delete previous character |
| ctrl-j | Create a new line |
| ctrl-k | Delete after current position in command |
| ctrl-n | Next command |
| ctrl-o | Execute current command and retrieve next command |
| ctrl-p | Previous command |
| ctrl-r | Retrieve command by search condition |
| ctrl-u | Delete command |
| ctrl-w | Delete before current position in command |

The arrow keys can be used for command retrieval and for positioning the cursor within the line.

2.11 On-line documentation

In the `~MIMERADM/dat` directory, some PDF-files are available for on-line display of the documentation for the MIMER product. Together with these files a plain text `readme_doc.txt` file exists, which gives some information on how to use the files.

3 MODULE SPECIFIC INFORMATION

3.1 MIMER/DB

This module includes the relational database manager in both single- and multi-user forms, an SQL compiler, etc. All database handling in all other MIMER modules is done through MIMER/DB.

3.1.1 Accessible files in MIMER/DB

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

Executable files:

| | |
|----------------------------|---|
| ~MIMERADM/bin/conv57 | Conversion utility for converting version 5 databanks to version 7. |
| ~MIMERADM/bin/dbc | Database Check program. Can be used to check if a databank is physically damaged. |
| ~MIMERADM/bin/dbextend | Databank extend utility. Can be used when a databank file exceeds the UNIX system ulimit. Must be run by the superuser (see the <i>MIMER Installation Guide for UNIX</i>). |
| ~MIMERADM/bin/ixchecks | Program to analyze the correctness of indexes. |
| ~MIMERADM/bin/mimtic | Script that installs the MIMER supplied terminfo definition files (see the <i>MIMER Installation Guide for UNIX</i>). |
| ~MIMERADM/bin/multiadm | Main script for the MULTIADM menu system interface (see the <i>MIMER Administration Guide for UNIX</i>). |
| ~MIMERADM/bin/multiinstall | Script that installs or updates the database environment for a MULTIADM user (see the <i>MIMER Administration Guide for UNIX</i>). |
| ~MIMERADM/bin/netsrv | The MIMER network server program, initiated by the UNIX administrator, multi-user mode (see the <i>MIMER Installation Guide for UNIX</i>). |
| ~MIMERADM/bin/sdbgen | System databank generation program for the DB module. |

| | |
|-----------------------------|--|
| ~MIMERADM/bin/singleinstall | Script that installs or updates a single-user database environment for any user. |
| ~MIMERADM/bin/upgr73 | Program to upgrade a version 7.1 or version 7.2 database to version 7.3. |

Library files:

As a guideline when linking a MIMER based application, dynamic or static, the example makefile installed as ~MIMERADM/dat/ex_makefile can be used.

One of the following shared libraries must be added to the link specification for each MIMER based application program, linked as a dynamic module. The file extension used for a shared library differs between different UNIX implementation. A frequently used extension is "so" which is used here as an example. The installed libraries have the correct extension for the specific environment in use.

| | |
|----------------------------|--|
| ~MIMERADM/lib/libmimer.so | DB multi-user runtime shared library. |
| ~MIMERADM/lib/libmimers.so | DB single-user runtime shared library. |

All of the following libraries must be added to the link specification for each MIMER based application program, linked as a static module.

| | |
|---------------------|--|
| ~MIMERADM/lib/db.a | DB common routines runtime library. |
| ~MIMERADM/lib/lr.a | MIMER Library Routines (Lower level). |
| ~MIMERADM/lib/lru.a | MIMER Library Routines (Upper Level). |
| ~MIMERADM/lib/mdr.a | MIMER Machine Dependent Routines (lowest level). |

In addition to the libraries listed above, one of the following two libraries must be included. The MIMER/DB application will become either single-user or multi-user depending on the library chosen.

| | |
|----------------------|---------------------------------|
| ~MIMERADM/lib/dbkm.a | DB multi-user runtime library. |
| ~MIMERADM/lib/dbks.a | DB single-user runtime library. |

3.1.2 Using singleinstall

This program performs the databank generation primarily for a single-user database, but the generated system may also be started as a multi-user system.

The name of the generated database must be included in the `/etc/sqlhosts` file. It may however be convenient to have the database `SINGLE` pointing to “.” (current directory) available in the `/etc/sqlhosts`. The `MIMER_DATABASE` can then be set to `SINGLE` and thus the database found in the current directory will be accessed.

For detailed information see chapter 6.

3.1.3 Exit routines

MIMER/DB version 7 does not support the use of the `exit1` exit routine by user application programs. Currently use of `exitc1` can be forced as an option, but it is not recommended. Refer to chapter 5 for a discussion of this feature.

The exit routines available for MIMER/DB (only for single-user mode) are located under the `~MIMERADM/lib` directory, and should be administered by the MIMERADM user.

3.2 MIMER/ESQL

The ESQL (Embedded SQL) module provides the ability to embed SQL statements in an ordinary programming language (currently only C is supported in the Unix environment). By the use of pre-processors provided in this module, the embedded program is converted to a pure host language program that can be compiled by the ordinary language compiler.

3.2.1 Accessible files in MIMER/ESQL

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

Executable file:

| | |
|---------------------------------|------------------------------------|
| <code>~MIMERADM/bin/esql</code> | ESQL pre-processor for embedded C. |
|---------------------------------|------------------------------------|

Data file:

| | |
|--------------------------------------|---|
| <code>~MIMERADM/dat/mimesql.h</code> | C-include file containing type definitions. |
|--------------------------------------|---|

Example files:

| | |
|---------------------------------------|--|
| <code>~MIMERADM/dat/example.ec</code> | Simple example program in embedded C. |
| <code>~MIMERADM/dat/dsql.ec</code> | Example routine library in embedded C using dynamic SQL. |
| <code>~MIMERADM/dat/dsql.h</code> | Header file for <code>dsql.ec</code> . |
| <code>~MIMERADM/dat/dsqlsamp.c</code> | Example program in C that calls the routines in <code>dsql.ec</code> . |

| | |
|---------------------------|--|
| ~MIMERADM/dat/blobsamp.ec | Example program in C that shows the use of the VARCHAR datatype to store BLOB's in a MIMER database. |
|---------------------------|--|

3.2.2 UNIX compilers supported

The output generated by the ESQL pre-processor can be compiled by the C compiler defined for the MIM_CC symbol in the ~MIMERADM/dat/makeopt file. Other language compilers from other software distributors may or may not be able to compile the ESQL output.

3.2.3 ESQL command syntax

An embedded C file may be preprocessed by issuing the following command:

```
$ esql infile.ec outfile.c
```

The ESQL pre-processor will read infile.ec, substitute all embedded SQL syntax with calls to the MIMER/DB library, and write the resulting program on outfile.c. Any SQL syntax errors found by ESQL will be reported on the terminal and in the generated output file.

The ESQL pre-processor recognizes the option -s, which suppresses the reporting of syntax errors on the terminal. Example:

```
$ esql -s infile.ec outfile.c
```

The exit value returned from ESQL is the number of errors found. If no errors are found, zero is returned (i.e. a successful completion).

More general information can be found in the *MIMER/SQL Reference Manual* and the *MIMER/SQL Programmers Manual*.

3.2.4 Compiling the example.ec program

The example.ec program is a very simple C program that illustrates how an embedded C program can be constructed. The program is written according to the X/Open XPG3 standard so it should be portable to any database manager that supports that standard.

The program logs in on the default database with user SYSADM password SYSADM (edit the program to change username or password). It then prints a list of all tables in the system by selecting all tuples from the X/Open standard view INFORMATION_SCHEMA.TABLES.

The program can be compiled by using the example makefile as follows (avoid trailing spaces when updating the **MYPROG** parameter in **makefile**):

```
$ mkdir example # Do everything in a sub directory
$ cd example
$ cp ~MIMERADM/dat/ex_makefile ./makefile
$ cp ~MIMERADM/dat/example.ec .
$ vi makefile
#
# The following parameter should be changed:
MYPROG = example
#
$ make
$ MIMER_DATABASE= (enter a database here)
$ export MIMER_DATABASE
$ ./example
```

3.2.5 Compiling the dsqlsamp program

The dsql.ec example file contains a simple library that enables calling programs to specify SQL statements in C strings, compile and execute those statements and to retrieve any result sets. The file uses dynamic SQL capabilities as specified in international standards and should be portable to other database managers that supports those standards.

The dsqlsamp.c file contains a simple program that lets users enter any SQL command interactively. The result of the SQL statement is displayed directly on the terminal. (The program can be thought of as a simple BSQL program.)

By using the example makefile, the program can be compiled by using the following commands (avoid trailing spaces when updating the parameters in **makefile**):

```
$ mkdir dsql # Do everything in a sub directory
$ cd dsql
$ cp ~MIMERADM/dat/ex_makefile ./makefile
$ cp ~MIMERADM/dat/dsql* .
$ vi makefile
#
# Update the makefile in two places.
# The following parameters should be changed:
MYPROG = dsqlsamp
MYFUNCS = dsql.o
#
$ make
$ ./dsqlsamp
```

3.2.6 Compiling the blobsamp program

The blobsamp.ec file contains a simple program which demonstrates the use of the VARCHAR datatype to store BLOB's. The test copies the contents of any file into the database as a BLOB, then it does a SELECT to pick it up again and differs the result against the original file. The file used must be copied to the file name "blob.r" and two objects, the ident BLOB and his table BLOB, must be created using for example BSQL (see the information given in the **blobsamp.ec** file).

By using the example makefile, the program can be compiled by using the following commands (avoid trailing spaces when updating the **MYPROG** parameter in **makefile**):

```
$ mkdir blobsamp          # Do everything in a sub directory
$ cd blobsamp
$ cp ~MIMERADM/dat/ex_makefile ./makefile
$ cp ~MIMERADM/dat/blobsamp.ec .
$ vi makefile
#
# The following parameter should be changed:
MYPROG = blobsamp
#
$ make
$ ./blobsamp
```

3.2.7 File handling conventions

The suffix **.ec** is normally used for an embedded C source file. Rules are included in the example makefile (**~MIMERADM/dat/ex_makefile**) that recognizes this suffix.

3.2.8 Type definitions

The file **~MIMERADM/dat/mimesql.h** contains type definitions that are included and used in generated C code.

3.3 MIMER/FMD

MIMER/FMD is only included in this version to support older application programs, and should not be used for development of new applications.

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 translate forms from the older screen interface, SH (Screen Handler), to FM forms.

3.3.1 Accessible files in MIMER/FMD

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

Executable files:

| | |
|----------------------|---|
| ~MIMERADM/bin/fmem | FM screen editor in multi-user mode. |
| ~MIMERADM/bin/fmes | FM screen editor in single-user mode. |
| ~MIMERADM/bin/fmshcm | FM/SH translation compiler, multi-user mode. |
| ~MIMERADM/bin/fmshcs | FM/SH translation compiler, single-user mode. |

Data file:

| | |
|-------------------------|-----------------------------|
| ~MIMERADM/dat/sysfm.exp | SYSFM databank import file. |
|-------------------------|-----------------------------|

3.3.2 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 filename is the same as the form name, with file suffix **.mpr** added.

3.3.3 FMD keypad usage

The FM editor uses the following function keys:

| | | | |
|----|------------|-----|------------|
| F1 | unassigned | F8 | unassigned |
| F2 | unassigned | F9 | unassigned |
| F3 | unassigned | F10 | unassigned |
| F4 | LEFT | F11 | unassigned |
| F5 | unassigned | F12 | HELP |
| F6 | RIGHT | F13 | RESET |
| F7 | unassigned | F14 | EXIT |

Other functions: CLEAR, DEL CHAR, INS/OVER, SEND/ENTER

See the *MIMER Release Notes for UNIX* for keypad layouts and details of the terminal key assignments.

3.4 MIMER/FMR

MIMER/FMR is only included in this version to support older application programs, and should not be used for development of new applications.

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

3.4.1 Accessible files in MIMER/FMR

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

| | |
|-----------------------|----------------------------------|
| ~MIMERADM/lib/fmlib.a | FM runtime library. |
| ~MIMERADM/lib/fmshi.a | FM/SH interface runtime library. |

3.4.2 Supported terminal types

The MIMER terminal interface uses the UNIX facility called **CURSES** to communicate with terminals. Because of this, MIMER/FM can support any terminal that **CURSES** supports.

The capabilities of terminals which are to be supported by the **CURSES** facility must be described in the UNIX terminal information (terminfo) database. It should be noted that there are large differences in the capabilities of different terminals. The FM runtime routines make use of standard capabilities (such as function keys) which are most often *not* defined in the terminal description supplied by the operating system (i.e. the terminal manufacturer). In addition, the manufacturer supplied descriptions for even relatively common terminals often contain errors. Although terminals not listed may work with FM if the corresponding terminal description is correct and sufficiently complete, Sysdeco Mimer AB only explicitly supports those terminals listed in the *MIMER Release Notes for UNIX*.

As usual with CURSES-based applications, the environment variables TERM and TERMINFO can be used to specify the TERM definition to be used for the current session.

3.4.3 Keypad mode

The terminal keypad mode (numeric or application) cannot be changed dynamically by the application program. However, MIMER/FM applications can simulate a numeric keypad by intercepting the key codes.

3.4.4 Linking applications to MIMER/FMR

Applications which use MIMER/FMR should add the FM runtime library to the link specification. (The file `~MIMERADM/dat/ex_makefile` is a makefile example).

The SH/FM-interface maps all MIMER/SHR routines to MIMER/FMR functions. Most (but not all) of the functionality of MIMER/SHR is supported. This interface is included in the FM/SH interface runtime library which must be linked in before the FM runtime library if a MIMER/SH application is going to use MIMER/FM instead.

3.5 MIMER/ISQL

The ISQL (Interactive SQL) module provides facilities for writing and executing SQL statements using a full screen editor. BSQL (Batch SQL) allows the user to execute SQL commands read from an input file (or the terminal). The COMMAND command is available to execute UNIX commands.

3.5.1 Accessible files in MIMER/ISQL

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

Executable files:

| | |
|---------------------|---|
| ~MIMERADM/bin/bsqlm | Line oriented SQL program, multi-user mode. |
| ~MIMERADM/bin/bsqls | Line oriented SQL program, single-user mode. |
| ~MIMERADM/bin/isqlm | Interactive SQL program (forms driven), multi-user mode. |
| ~MIMERADM/bin/isqls | Interactive SQL program (forms driven), single-user mode. |

Data files:

| | |
|----------------------------|-------------------------------|
| ~MIMERADM/dat/syshelp.exp | SYSHELP databank import file. |
| ~MIMERADM/dat/examples.dat | SQL statement examples. |

3.5.2 Keyboard interrupt

BSQL allows the user to interrupt the execution of commands by pressing the interrupt key (often set to **ctrl-c**). This will take the user to the BSQL prompt as soon as the current database request, if any, is completed.

Note! Pressing the quit key (often set to **ctrl-**) will terminate BSQL and return the user to the UNIX command level. A file called **core** is also created on the current working directory and should be removed.

3.5.3 ISQL keypad usage

ISQL uses the following function keys:

| | | | |
|----|---------------|-----|--------------|
| F1 | REMOVE | F8 | UP 15 |
| F2 | DOWN 15 | F9 | JOIN COMMAND |
| F3 | EOL | F10 | EXECUTE |
| F4 | LEFT 80 | F11 | TOGGLE |
| F5 | UNDO | F12 | HELP |
| F6 | RIGHT 80 | F13 | RESET |
| F7 | SPLIT COMMAND | F14 | EXIT |

Other functions: CLEAR, DEL CHAR, INS/OVER, SEND/ENTER

See the *MIMER Release Notes for UNIX* for keypad layout and details of the terminal key assignments.

3.5.4 The MIMINI initiation file

The initiation file MIMINI can be used to store login procedures for both ISQL and BSQL.

3.6 MIMER/PGD

MIMER/PGD is only included in this version to support older application programs, and should not be used for development of new applications.

The PGD (Program Generator Development) module is used to develop application programs. The application programs are written using PG commands and can be run interactively. After the programs have been tested, C code can be generated from the PG code for improved performance.

The COMMAND command is available to execute UNIX commands.

3.6.1 Accessible files in MIMER/PGD

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

Executable files:

| | <u>Screen I/O</u> | <u>LISP</u> | <u>Mode</u> |
|----------------------|--|-------------|-------------|
| ~MIMERADM/bin/pgim | FM | interpreted | Multi |
| ~MIMERADM/bin/pgis | FM | interpreted | Single |
| ~MIMERADM/bin/pgm | FM | compiled | Multi |
| ~MIMERADM/bin/pgs | FM | compiled | Single |
| ~MIMERADM/bin/pgshim | SH | interpreted | Multi |
| ~MIMERADM/bin/pgshis | SH | interpreted | Single |
| ~MIMERADM/bin/pgshm | SH | compiled | Multi |
| ~MIMERADM/bin/pgshs | SH | compiled | Single |
| ~MIMERADM/bin/cgen | Used by PG to produce C-language code. | | |
| ~MIMERADM/bin/pgrun | Command script to link simple PG applications. | | |

The “Mode” column above indicates whether the program is linked in single-user or multi-user mode. The “Screen I/O” column indicates the screen handler that is used if programs are executed interactively within MIMER/PG. Finally, the “LISP” column indicates whether the LISP part of MIMER/PG is compiled or interpreted. A compiled version executes faster but needs more memory. The interpreted version uses less memory, but needs more CPU power to do the same job.

3.6.2 Functionality of the code generator

The code generator, **cgen**, is started by the GENERATE command, available in MIMER/PG. The PG prompt returns when the generation is complete. The generating process executes with a low priority to provide the best possible response for interactive users. The default is a UNIX **nice** value of 10, but if the environment variable MIMER73_NICE is set, the value given is used for nice. The file **gen.log** is created in the current working directory (or in a directory in the MIMER73_PATH environment variable, if one exists) during the generation process. If the **cgen** executable is not found through the UNIX environment variable PATH a lisp file is created, but please note that no error code is given in this case. When the lisp file is created, the cgen utility can be run manually to complete the code generation.

Note! Be sure that your PATH environment variable does not reference a directory containing executables for an older MIMER version, because **cgen** has the same name in older MIMER versions.

3.6.3 File handling conventions

Input

Sequential files containing PG statement definitions are accessed via the LOAD FILE command which uses the filename specified. Contrary to the instructions in the *MIMER/PG Reference Manual*, the file must be specified in double quotes in the UNIX environment. To comply with MIMER conventions, it is recommended that PG source filenames end with the suffix **.pg**, for example:

```
PG>LOAD FILE "appl.pg" ;
```

If a MIMER73_PATH environment variable local to the user program is present and relative filenames are used, it will be used to help locate the input files.

Output

Output files created by the OUTPUT command are created using exactly the name specified. If relative filenames are used, the MIMER73_PATH environment variable will affect the final location of the file.

Application program source files produced by the GENERATE command in PG have the same name as the PG program or procedure, but with an appropriate suffix appended (i.e. **.c** for C language).

3.6.4 The pgrun command script

There is a command script named **pgrun** in the **~MIMERADM/bin** directory. This script can be used to compile and link simple PG applications. Complicated applications (with several separately compiled procedures, and/or using external libraries) cannot be linked by **pgrun**. When such applications are to be compiled and linked, the user should write his own compile/link specification using the example makefile in the **~MIMERADM/dat** directory as a guide.

Pgrun takes a file generated by MIMER/PG and assumes that the program name is the same as the filename. The **pgrun** script checks the filename and generates a main-function if necessary. This is transparent to the user.

Example:

In the example listed below, pgrun is used to compile and link a PG-generated application named **pgappl**. To instruct PG to generate C-code do the following:

```
$ pgrun
File generated by MIMER/PG      [main.c]      : pgappl.c
Single or multi mode (s,m)      [m]          : <return>
MIMER/FM or MIMER/SH (fm,sh)    [fm]       : sh
MIMER/SH coupling file          [couple.c]   : shpict.c
Compile with optimization (y/n)  [n]         : <return>
compiling pgappl.c ...
```

The command script first asks for the name of the generated application file (**pgappl.c**), if MIMER/DB should be used in single- or multi-user mode (default: multi-user) and which 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. Optimization is only recommended for relatively small PG programs or if performance is a very high priority because optimizing large files drastically increases compiling time. If the RETURN key is pressed, the default value shown inside the brackets will be used. When the procedure is completed, the application can be executed.

```
$ pgappl
```

3.6.5 Keyboard interrupt

PG allows the user to interrupt the execution of commands by pressing the interrupt key (often set to **ctrl-c**). This will take the user to the PG prompt.

Note! Pressing the quit key (often set to **ctrl-^**) will terminate PG and return the user to the UNIX command level. A file called **core** is also created on the current working directory and can be removed.

If a keyboard interrupt is reported during generation of a PG program or procedure, this usually indicates that the PG code generator has run out of memory. Either build a larger version of the PG executable (see the *MIMER Installation Guide for UNIX*), or break the PG code into smaller units.

3.6.6 The MIMINI and PG initiation files

The initiation 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 in the current device and directory.

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 MIMER/PGR

MIMER/PGR is only included in this version to support older application programs, and should not be used for development of new applications. For new applications, use MIMER/ESQL or ODBC.

This module includes the runtime libraries needed to link PG applications.

3.7.1 Accessible files in MIMER/PGR

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

Library files:

| | |
|-----------------------|---|
| ~MIMERADM/lib/pgfml.a | PG screen handler interface for MIMER/FM |
| ~MIMERADM/lib/pgl.a | PG runtime library. |
| ~MIMERADM/lib/pgshl.a | PG screen handler interface for MIMER/SH. |

Data file:

| | |
|-------------------------|-----------------------------|
| ~MIMERADM/dat/syspg.exp | SYSPG databank import file. |
|-------------------------|-----------------------------|

3.7.2 Linking a PG application

There are three runtime libraries for MIMER/PG:

- the main PG runtime library
- the PG/FM screen handler interface library
- and the PG/SH screen handler interface library.

All PG applications must link to the main PG runtime library and to one of the libraries containing the screen handler interface, depending on which screen I/O interface is being used. If no screen I/O is used in the application, the PG/FM interface library should be used.

The example makefile `~MIMERADM/dat/ex_makefile` can be used as a template for a makefile to link PG applications.

Note! When linking with PG/SH interface library, the coupling file generated by the SH compiler should be listed before the library in the link specification. The coupling files are described in the *MIMER/SH Reference Manual*.

The `pgrun` command script can be used when linking simple PG-applications (see the PGD module).

3.8 MIMER/PI

MIMER/PI is the “level 4” Programming Interface provided with MIMER version 4. It is only included in this version to support older application programs, and should not be used for development of new applications. For new applications, use MIMER/ESQL or ODBC.

The PI module also contains the CL (Command Language) program, for compatibility with CL command files written for version 4 of MIMER. The replacement for this module is MIMER/BSQL, which should be used in new applications.

3.8.1 Accessible files in MIMER/PI

The PI module contains the following files that are available to the user:

Executable files:

| | |
|-------------------|----------------------------------|
| ~MIMERADM/bin/clm | The CL program, multi-user mode |
| ~MIMERADM/bin/cls | The CL program, single-user mode |

Library file:

| | |
|--------------------|----------------------------------|
| ~MIMERADM/lib/pi.a | The PI interface runtime library |
|--------------------|----------------------------------|

3.8.2 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 other libraries.

Note: Although the PI library is functionally equivalent to older versions of the PI library, from the user's viewpoint, the old distribution of the library *cannot* be used when linking applications in the version 7 environment.

3.8.3 Exit routines

This version of MIMER/PI supports the PI (level 4) exit routine **exit4**, as documented in the *MIMER/DB version 4 Reference Manual*.

The exit routine **exitc4** is not supported.

3.9 MIMER/QF

MIMER/QF is only included in this version to support older application programs, and should not be used for development of new applications.

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

3.9.1 Accessible files in MIMER/QF

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

Executable files:

| | |
|-------------------|-------------------------------------|
| ~MIMERADM/bin/qfs | The QF program in single-user mode. |
| ~MIMERADM/bin/qfm | The QF program in multi-user mode. |

Data file:

| | |
|-------------------------|-----------------------------|
| ~MIMERADM/dat/sysqf.exp | SYSQF databank import file. |
|-------------------------|-----------------------------|

3.9.2 QF keypad usage

QF uses the following function keys:

| | | | |
|----|------------|-----|------------|
| F1 | unassigned | F8 | UP |
| F2 | DOWN | F9 | unassigned |
| F3 | unassigned | F10 | unassigned |
| F4 | PREV APPL | F11 | unassigned |
| F5 | unassigned | F12 | HELP |
| F6 | NEXT | F13 | unassigned |
| F7 | unassigned | F14 | EXIT |

Other functions: CLEAR, DEL CHAR, INS/OVER, SEND/ENTER

See the *MIMER Release Notes for UNIX* for keypad layout and details of the terminal key assignments.

3.10 MIMER/QL

The QL (Query Language) module gives the user the ability to manipulate tables by using QL commands. This module is primarily intended to provide compatibility with older versions of MIMER. The COMMAND command is available to execute UNIX commands.

3.10.1 Accessible files in MIMER/QL

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

Executable files:

| | |
|---------------------|---|
| ~MIMERADM/bin/qlm | The QL program with FM interface, multi-user mode. The QF- and RG-modules are also linked into this program. |
| ~MIMERADM/bin/qls | The QL program with FM interface, single-user mode. The QF- and RG-modules are also linked into this program. |
| ~MIMERADM/bin/qlshm | The QL program with SH interface, multi-user mode. |
| ~MIMERADM/bin/qlshs | The QL program with SH interface, single-user mode. |

Library file:

| | |
|--------------------|---------------------------|
| ~MIMERADM/lib/ql.a | MIMER/QL runtime library. |
|--------------------|---------------------------|

Data file:

| | |
|-------------------------|-----------------------------|
| ~MIMERADM/dat/sysql.exp | SYSQL databank import file. |
|-------------------------|-----------------------------|

3.10.2 Keyboard interrupt

QL allows the user to interrupt the execution of commands by pressing the interrupt key (often set to **ctrl-c**). This will take the user to the QL prompt.

Note! Pressing the quit key (often set to **ctrl-^**) terminates QL and returns the user to the UNIX command level. QL must then be restarted from the beginning. A file called **core** is also created on the current working directory and should be removed.

3.10.3 QL keypad usage

QL reads and executes commands on a line by line basis and does not ordinarily employ full screen mode.

QL procedures may accept input via either the SH or FM modules, in which case the default keyboard mapping for the respective modules applies.

3.10.4 The MIMINI initiation file and QL

The initiation file MIMINI can 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 in the current device and directory.

3.11 MIMER/RG

MIMER/RG is only included in this version to support older application programs, and should not be used for development of new applications.

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 over the layout. The RGF utility can generate RGL code which then may be modified.

The COMMAND command is available to execute UNIX commands.

3.11.1 Accessible files in MIMER/RG

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

Executable files:

| | |
|----------------------|---|
| ~MIMERADM/bin/rgbatm | The RG batch (line oriented interface) program, multi-user mode. |
| ~MIMERADM/bin/rgbats | The RG batch (line oriented interface) program, single-user mode. |
| ~MIMERADM/bin/rgm | The RG interactive (forms driven) program, multi-user mode. |
| ~MIMERADM/bin/rgs | The RG interactive (forms driven) program, single-user mode. |

Library file:

| | |
|--------------------|--|
| ~MIMERADM/lib/rg.a | RG runtime library for RG user applications. |
|--------------------|--|

Data file:

| | |
|-------------------------|-----------------------------|
| ~MIMERADM/dat/sysrg.exp | SYSRG databank import file. |
|-------------------------|-----------------------------|

3.11.2 RG keypad usage

RG uses the following function keys:

| | | | |
|----|-----------------|-----|----------------|
| F1 | unassigned | F8 | UP (PAGE BKWD) |
| F2 | DOWN (PAGE FWD) | F9 | unassigned |
| F3 | unassigned | F10 | unassigned |
| F4 | unassigned | F11 | unassigned |
| F5 | unassigned | F12 | HELP |
| F6 | unassigned | F13 | unassigned |
| F7 | unassigned | F14 | EXIT |

Other functions: CLEAR, DEL CHAR, INS/OVER, SEND/ENTER

See the *MIMER Release Notes for UNIX* for keypad layout and details of the terminal key assignments.

3.11.3 The MIMINI initiation file and RG

The initiation file MIMINI can be used to store login procedures for RG.

The MIMINI file can also include commands for execution by the RGBATCH function (described in the *MIMER/RG Reference Manual*).

3.11.4 Linking RG applications

Programs calling RG functions through the RG programming interface must be linked with the RG runtime library by adding it's name to the link specification. The file `~MIMERADM/dat/ex_makefile` gives an example of linking an RG application.

3.12 MIMER/SHD

MIMER/SHD is only included in this version to support older application programs, and should not be used for development of new applications.

The SHD (Screen Handler Development) 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 asynchronous and synchronous terminal types, the FM (Forms Manager) should be used for screen management.

The MIMER_HOME environment variable need to be correctly defined for the proper behavior of this module.

3.12.1 Accessible files in MIMER/SHD

The SHD module includes the following files that are available to users:

Executable files:

| | |
|--------------------|------------------------------|
| ~MIMERADM/bin/shem | SH editor, multi-user mode. |
| ~MIMERADM/bin/shes | SH editor, single-user mode. |

Data files:

| | |
|--------------------------|--|
| ~MIMERADM/dat/syssh.exp | SYSSH databank import file. |
| ~MIMERADM/dat/shesou.shf | SH picture file for the SH editor. It is handled automatically by the screen editor. |

3.12.2 Terminal specific key sequences

Below is a list of the default keyboard mappings for SH under UNIX. This keyboard mapping does not follow the recommended keyboard mapping for MIMER applications on UNIX. Instead, it is intended to be closely compatible with the mapping used with SH on other operating systems and earlier versions of MIMER. Almost any correct TERMINFO terminal descriptions supplied with UNIX should work with SH.

On UNIX the SH terminal interface is mapped towards the CURSES screen handling and optimization package. The CURSES interface layer of MIMER converts the terminal key codes into escape sequences interpreted by SH. The mapping between this layer and SH is made in the **trmnew.def** file supplied under the **~MIMERADM/dat** directory. The terminal chosen for this "internal mapping" is VT100. Any other terminal description within the **trmnew.def** file is invalid, and should not be used from SH. Instead use the TERM environment variable to select the terminal type. CURSES evaluates the TERM variable and then handles all terminal interaction according to the setting.

By default four function keys are mapped in the **trmnew.def** file. These apply to the function keys F11-F14 as shown in the terminal key map layout in the *MIMER Release Notes for UNIX*. It looks as follows in the **trmnew.def** file (for the VT00 terminal definition):

```
!
In_func_key01      : ESC O P
In_func_key02      : ESC O Q
In_func_key03      : ESC O R
In_func_key04      : ESC O S
!
```

It is, however, possible to map 14 function keys to MIMER/SH. Suitable F1-F14, according to the terminal key map layout shown in the *MIMER Release Notes for UNIX* are mapped by replacing the section shown above with the following section in the **trmnew.def** file (for the VT100 terminal definition):

```
!
In_func_key01      : ESC O q
In_func_key02      : ESC O r
In_func_key03      : ESC O s
In_func_key04      : ESC O t
In_func_key05      : ESC O u
In_func_key06      : ESC O v
In_func_key07      : ESC O w
In_func_key08      : ESC O x
In_func_key09      : ESC O y
In_func_key10      : ESC O p
In_func_key11      : ESC O P
In_func_key12      : ESC O Q
In_func_key13      : ESC O R
In_func_key14      : ESC O S
!
```

The functions listed below are identified by symbolic names as given in the *MIMER/SH Reference Manual*.

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. Also note that MIMER/SH requires an escape key (if none exists, one must be added to the terminal setup).

General control keys (if defined in TERMINFO for your terminal):

| Name | Key Sequence | Description |
|----------|--------------|--|
| <Leave> | @ | Update current definition and terminate the editor session |
| <Abort> | ! | Leave current definition without updating information and reinstate the previous definition |
| <Update> | \$ | Update current definition |
| <EntRef> | & | Enter reference definition for update. This is only valid if positioned in a field that refers to another definition |

Control keys for MIMER/SH (SH-editor):

| Name | Key Sequence | Description |
|-------------|---------------------|--|
| <BegOLine> | ctrl-a | Jump to beginning of line |
| <EndOLine> | ctrl-e | Jump to end of line |
| <RfrScr> | ctrl-r | Refresh screen |
| <InsBLine> | ctrl-l | Insert a blank line |
| <UnDLine> | ctrl-u | Undelete previously deleted line |
| <DelELine> | ctrl-k | Delete to end of line |
| <DelCLine> | esc <cr> | Delete current line (and add blank at end of screen) |
| <SplLine> | esc / | Split line at cursor |
| <DelPChar> | del | Delete preceding character |
| <DelCChr> | ctrl-d | Delete current character |
| <UppCase> | esc u | Convert word to upper case |
| <LowCase> | esc l | Convert word to lower case |
| <CapCase> | esc c | Capitalize word |
| <Over/Ins> | esc t | Overstrike/insert mode toggle |
| <EntCom> | esc k | Enter command mode |
| <PFDef> | esc q | Define picture field at current position |
| <PFTag> | esc . | Set tag on picture field at current position |
| <PFMove> | esc m | Move tagged picture field to current position |
| <PFNext> | esc n | Jump to next defined field in picture |

3.13 MIMER/SHR

MIMER/SHR is only included in this version to support older application programs, and should not be used for development of new applications.

This module contains the runtime package for MIMER/SH, a screen handler for asynchronous terminals. The SHR module contains the **shc** executable, which is a screen compiler.

Applications requiring portability between synchronous terminals, block mode terminals, and distributed processing environments should use FM (Forms Manager) for screen management.

Under UNIX, SH uses the same terminal support routines as FM. Therefore terminals which are supported for FM are also supported for SH.

The MIMER_HOME environment variable need to be correctly defined for the proper behavior of this module.

3.13.1 Accessible files in MIMER/SHR

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

Executable files:

| | |
|---------------------|------------------------------|
| ~MIMERADM/bin/shc | SH Compiler utility. |
| ~MIMERADM/bin/termd | Terminal definition utility. |

Library file:

| | |
|-----------------------|---------------------|
| ~MIMERADM/lib/shlib.a | SH runtime library. |
|-----------------------|---------------------|

Data files:

| | |
|--------------------------|--|
| ~MIMERADM/dat/shfunc.def | Function key definitions. |
| ~MIMERADM/dat/shlang.dat | Runtime error messages for SHC. |
| ~MIMERADM/dat/shtrmnew | Data file read by all SH applications. This file is generated by the TERMD program during the MIMER/SHR module installation phase. |
| ~MIMERADM/dat/trmnew.def | Input file to the TERMD program that generates the shtrmnew file. |

3.13.2 Initiation files

The initiation file MIMINI can be used to store login procedures for SH. The default name for the file is MIMINI in the current device and directory.

3.13.3 Terminal specific key sequences

The keyboard mapping described for SHD above also applies to SHR.

3.13.4 The SH compiler

The MIMER/SH compiler is started with the command:

```
$ shc
```

Note that the compiler is a separate utility program which operates directly on the picture definition source file, independent of the MIMER installation. There is no MIMER 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 filenames. It is suggested that the following file suffixes are added to the end of the names:

- Picture definition source file: **.sou**
- Picture definition object file: **.shf**
- Coupling source file: **.c**

No suffixes will be added automatically.

The operator can define the minimum size of the picture definition object file if desired. This size is given in MIMER pages (1 MIMER page = 4 UNIX blocks or 2048 bytes). 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 minimize the risk of object file fragmentation.

The coupling file is generated in the C-language.

Example

```
$ shc
Picture definition source file      : hotelpic.sou
Picture definition object file     : hotelpic.shf
Object file size                   : 5
Generated coupling source file     : hotelpic.c
```

WARNING! MIMER under UNIX does *not* allow the same filename to be used as both the file input and output name. If the same filename is used, it may result in the silent destruction of the input file and/or an error message reporting that the file is "locked by another user".

3.13.5 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 is not to be linked with the application. The shlib.a runtime library for SH routines should be added to the list of libraries in the link specification.

Example

Link an application called **hotappl** with a coupling file **hotelpic.c** for multi-user operations (remember that the C-language coupling file must first be compiled with the C compiler):

```
$ cc hotelpic.c
```

Then create your own copy, suitable called **makefile**, of the example makefile found as `~MIMERADM/dat/ex_makefile`, edit it, and then run **make**. Note that the C compiler requires that C source files have **.c** as a suffix.

When the SH application is started, it will use the `MIMER73_PATH` environment variable to find any necessary picture definition files. The current directory must be included if it is to be searched for picture definition files. If no `MIMER73_PATH` variable is defined, the current directory is the only one searched.

3.13.6 Runtime version check facility

A consistency check between the SH software version and picture definition input data is made at runtime for SH applications.

Inconsistencies between the SH software version and the version of the picture definitions generates the following error message:

```
Version error, definition file
```

Picture definition version inconsistencies indicate either that the picture definitions were compiled with an incorrect version of the SH compiler, or that the application is linked with an incorrect runtime library. The error is corrected either by recompiling the picture definitions or relinking the application.

3.14 MIMER/UTIL

The UTIL program provides several functions required to maintain and create a database: Backup/Restore, Statistics, Read Log, Export/Import and Shadowing.

3.14.1 Accessible files in MIMER/UTIL

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

Executable files:

| | |
|-------------------------|---|
| ~MIMERADM/bin/dbopenm | A program that opens all available databanks, multi-user mode. |
| ~MIMERADM/bin/dbopens | A program that opens all available databanks, single-user mode. |
| ~MIMERADM/bin/mimchop | Program that removes line separators from a standard UNIX text file and pads the lines to a stream of fixed 80 byte records. The resulting format is called "Unformatted, fixed record length". |
| ~MIMERADM/bin/mimunchop | Program that reads a stream of fixed 80 byte records, and converts them to a standard UNIX text file. This program performs the reverse operation of the mimchop program. The resulting format is called "Formatted, variable record length". |
| ~MIMERADM/bin/utills | The UTIL program in single-user mode. |
| ~MIMERADM/bin/utilm | The UTIL program in multi-user mode. |

Data file:

| | |
|--------------------------|------------------------------|
| ~MIMERADM/dat/sysmsg.exp | SYSMSG databank import file. |
|--------------------------|------------------------------|

3.14.2 File handling conventions

Under UNIX, all filenames should be fully specified. No default suffixes are added or required by any of the utility functions. It is recommended that the suffix **.exp** is used for files to be read or written by the import/export function, but this is not essential.

MIMER/UTIL uses files with fixed record length (80) for the export/import functions. This means that files where records (lines) are separated with new line characters (which is normal under UNIX) have to be converted. The **mimchop** program can do this.

Note! If the file is formatted, the message "Unexpected end of file encountered" will be issued!

3.14.3 Using **mimchop**

The **mimchop** program reads a number of lines from the standard input device. Each line is terminated by a newline character (as is normal for text files under UNIX). The program writes the lines to the standard output device with no line separators. All lines are padded with blank characters (or truncated) to a fixed 80 character record length.

This program is useful, since the import/export function uses files with 80 byte fixed length records.

This is an example of how the program can be used:

```
$ mimchop < formatted_file > fixed_rec_file
```

This command will copy the file **formatted_file** to **fixed_rec_file**, removing newline characters and making fixed length records in the process.

3.14.4 Using **mimunchop**

This program performs the reverse of **mimchop**. It reads a stream of 80 byte fixed length records from standard input, and writes lines terminated by a newline character to standard output. For example:

```
$ mimunchop < fixed_rec_file > formatted_file
```

This command copies the fixed record file **fixed_rec_file** to **formatted_file**. Please note that since the **mimunchop** program does not remove trailing blank characters from each line, passing a UNIX file through **mimchop** and then through **mimunchop** does not exactly reproduce the original file!

3.14.5 The MIMINI initiation file

The initiation file MIMINI can be used to store login procedures for MIMER/UTIL. The default name for the file is MIMINI in the current device and directory.

4 DATA TYPES USED IN MIMER/DB

4.1 Internal DB representation

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

The character data type uses the ISO8859-1 standard (see the *MIMER/SQL Reference Manual, Appendix B*). This standard specifies graphic characters and the representation of each character.

Note: If an application is run on different machines which use different character representations, MIMER will use the ISO character standard when transferring data between them. If an application only runs on machines with the same character representation, the bytes entered and output will always be the same.

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

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:

- **Fixed length character**
This data type contains the number of bytes (characters) specified as the length of the variable of this type.
- **NULL terminated character string**
This data type, which is consistent with the C-language string data type, contains a number of bytes followed by, and terminated with, a byte with the value of NULL. In a NULL byte all bits are off (binary zero).
- **Floating point**
Depending on the length specified, this data type is implemented as either 4 bytes or 8 bytes.

- **Decimal**
This data type is a packed BCD number containing a maximum of 18 digits.
- **Small integer**
A signed integer consisting of two bytes.
- **Large integer**
A signed integer consisting of four bytes.

5 INSTALLING A SINGLE-USER DATABASE SYSTEM

Any UNIX user with access to the MIMER executable files can build and run a MIMER single-user system. The purpose of single-user systems is to provide an easy way for ordinary users to create a database for testing. Production databases should use multi-user mode.

An application that is linked in single-user mode includes all the database code in the image. Since the application accesses the databank files directly, only one program can access the databanks at a time.

Note: Since there is no database server started for a single-user system, the database cannot be accessed from remote machines using client-server techniques.

The single-user MIMER system requires a number of system databanks before it can be run successfully. These databanks are generated by a utility called **singleinstall**.

5.1 The installation procedure

Before running the **singleinstall** script, it is advisable to check that the environment variable PATH includes an entry to the MIMER executable files (including the **singleinstall** script). If not, the MIMER executables cannot be run without typing full command pathnames. For information on defining a new entry in a user PATH definition, read the UNIX system's manuals or consult the UNIX system administrator. After you are sure that the path to **~MIMERADM/bin** is included in the PATH environment, you may run the **singleinstall** script. Below is an example of running the script. The example is run with the MIMER_DATABASE environment variable set to "SINGLE". The name "SINGLE" is defined as a LOCAL database in the **/etc/sqlhosts** file associated with the directory path "." (current directory). Note that relative directories, such as ".", are only legal for single-user systems.

The `singleinstall` example:

```

$ MIMER_DATABASE=SINGLE; export MIMER_DATABASE
$ PATH="$PATH:/opt/products/mimer7"; export PATH
$ singleinstall
MIMER 7.3.1 SINGLE-USER SYSTEM INSTALLATION UTILITY

This utility will install a new MIMER single-user system (or
update a previously installed system). You are assumed to be
the user that will own and use the Mimer single environment
that will be created.

Note that you are responsible for your own login setup, which
must include a definition of the MIMER_DATABASE environment
variable. The database defined in MIMER_DATABASE must be
available as a LOCAL database in the /etc/sqlhosts file, where
the given path is the one that this utility will operate on.
You must also remember to add the MIMER bin directory path to
your own environment PATH variable, to be able to automatically
reach the Mimer programs.

You may run this script any number of times. It will not harm
a previously performed installation.

Do you want to continue? (y/n) [y] y

GENERATION OF MIMER VERSION 7.3.1 SYSTEM DATABANKS

Current MIMER installation is located on: /opt/products/mimer7

OK to continue? (y/n) [y] y

The MIMER system requires a system administrator password.
This user is identified by the username SYSADM (or sysadm)
and the password given below.

Type the SYSADM password: XXXXXX
Confirm the SYSADM password: XXXXXX

Databanks will be generated on the primary directory of the
search path defined in the /etc/sqlhosts file:
> .

After they have been generated, databanks can be moved to other
directories given in the /etc/sqlhosts search path.
In particular, it is recommended that the TRANSDB and LOGDB
databanks, be located on disks separate from each other and from
the rest of the system databank files.
Note that the multi-user system should be stopped when moving
databank files to other locations.
The search path for SINGLE is:
> .

Ok to continue? (y/n) [y] y

```

The following system databanks can be generated:

| Option | Databank(s) |
|--------|---------------------------|
| 1 | SYSDB/LOGDB/TRANSDB/SQLDB |
| 2 | SYSFM |
| 3 | SYSHELP |
| 4 | SYSMSG |
| 5 | SYSPG/WORKDB |
| 6 | SYSQF |
| 7 | SYSQL |
| 8 | SYSRG/SYSRGOUT |
| 9 | SYSSH/CMD |
| 10 | BOOKDB/HOTELDB/ROOMSDB |
| 0 | Exit |

Select databanks to create ("a" for all):

> a

The operations are logged on /tmp/alice.mimgen

```
SYSDB/TRANSDB/LOGDB/SQLDB ... generated in .
SYSFM ... generated in .
SYSHELP ... generated in .
SYSMSG ... generated in .
SYSPG/WORKDB ... generated in .
SYSQF ... generated in .
SYSQL ... generated in .
SYSRG/SYSRGOUT ... generated in .
SYSSH/CMD ... generated in .
BOOKDB/HOTELDB/ROOMSDB ... generated in .
```

Exiting database generation!

Database generation completed!

Type <return> to continue... <return>

\$

At this point, the selected databanks should be available and the single-user executables ready to use.

5.2 Installation errors

The **sdbgen** utility is used to create the system databanks (SYSDB, TRANSDB, LOGDB, SQLDB). Under certain circumstances **sdbgen** may fail, giving the error message:

```
*** Failed to create databank. Error: -16149 ***
Do you want to try again(<No>) ? <return>
*** Incomplete System Databanks created ***
```

This error message originates from the databank I/O routines and is machine specific. In UNIX systems it often can be explained by the fact that there are no free file locks available. The number of file locks available in the system is configurable in the UNIX kernel (see your local UNIX system documentation).

Before re-starting the installation procedure, remember to remove the old incomplete database, for which the creation was aborted (see error message above).

Note that similar error messages may occur in other situations for the same reason.

A EXIT ROUTINES

This section describes the use of the **exitc1** error handling routine for application programs using the LEVEL2 API.

The use of other exit routines to change sizes of internal areas is described in the *MIMER Administration Guide for UNIX*.

A.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 version 4 *MIMER/DB Reference Manual*). User-written **exitc1** routines could be linked with application programs for special error handling purposes.

In later MIMER versions, 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.

User-written exit routines are not supported from version 5 of MIMER because the standard DB runtime libraries will be implemented as a shared library in the future. To maintain compatibility with application programs written for MIMER version 4, that depend on user-written **exitc1** routines, support for **exitc1** is still present, but this may be removed in future versions.

A.2 Standard error handling

A.2.1 Using standard error handling

The **seter2** routine allows the user to set error handling to one of four standard modes, which covers program abort or continue, return error code and display error message (see the Low Level Interface section in the *MIMER Runtime Libraries Programmers Manual* for details).

It is recommended that application programs originally written in version 4 should be 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 practical. In this case, the application can be linked to a user-written **exitc1** routine.

A.2.2 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 implementations of this MIMER version)

B TERMINAL DEFINITIONS

B.1 Modules concerned

The following MIMER programs depend upon the MIMER/FM screen handler that uses the CURSES based MIMER terminal interface:

- MIMER/ISQL (Interactive SQL)
- MIMER/QF (Query-by-Forms)
- MIMER/RG (Report Generator)
- MIMER/FME (Forms Manager Editor)
- MIMER/QL (Procedures in the Query Language)
- MIMER/PG (Program Generator applications)

MIMER/SH uses the same terminal interface as MIMER/FM, and therefore the information given may be of interest for users of MIMER/SH, as well.

Chapter 3 of this guide gives a description of each module, and the keys it uses.

The list of supported terminal types for a specific platform is available in the *MIMER Release Notes for UNIX*.

B.2 Terminfo files

The MIMER terminal interface uses the UNIX CURSES package for screen handling and optimization. CURSES behaves on basis of the setting of the TERM environment variable, that should be set to the appropriate terminal type defined in the TERMINFO database.

In order to support the use of the keypad and special function keys, MIMER uses own terminal definitions. This is necessary since many original terminal definitions lacked appropriate support for function keys.

The new terminal definitions are stored with the same name as the original definition, but with an "-M" appended at the end of the name. Thus, the name for the MIMER-supplied definition of the **vt100** terminal is **vt100-M**.

The current setting of the environment variable TERM defines the terminal type. When deciding which terminal definition to use, the MIMER terminal interface first tries to append a "-M" to the terminal specified in the TERM variable. If this terminal definition is not found, the TERM variable definition is used unchanged. This means that if the current setting of the TERM variable is "vt100", the MIMER terminal interface will automatically use the MIMER-supplied definition called "vt100-M". If this definition is not found, the standard "vt100" definition is used, and the ability to use for example the function keys may be lost.

B.3 Common key definitions

MIMER/FM supports the following keys and key sequences, regardless of the terminal type selected. (Note, that all functions are not used in every MIMER program. Also note that the operating system **stty** setup may trap control sequences, before they reach MIMER):

| Key | Function |
|-------------|---|
| DOWN ARROW | Move cursor down (FM_MOVE_DOWN) |
| UP ARROW | Move cursor up (FM_MOVE_UP) |
| LEFT ARROW | Move cursor left (FM_MOVE_LEFT) |
| RIGHT ARROW | Move cursor right (FM_MOVE_RIGHT) |
| DEL | Delete previous character (FM_DELETE_PRECEDING_CHAR) |
| RETURN | Jump to the first field in the next row (FM_NEXT_LINE). In a field on the last line of the screen, RETURN has the same effect as ENTER i.e. initiates processing of the current form. |
| TAB | Jump to next field (FM_NEXT_FIELD) |
| <ESC> TAB | Jump to previous field (FM_PREVIOUS_FIELD) |
| CTRL E | Jump to end of field (FM_JUMP_TO_END_OF_FIELD) |
| CTRL G | Move to home position (FM_MOVE_HOME) |
| CTRL K | Clear to end of field (FM_CLEAR_TO_END_OF_FIELD) |
| CTRL L | Refresh screen (FM_REFRESH_SCREEN) |
| CTRL N | Move cursor down (FM_MOVE_DOWN) |
| CTRL P | Move cursor up (FM_MOVE_UP) |
| CTRL R | Refresh screen (FM_REFRESH_SCREEN) |
| CTRL T | Numeric/application keypad toggle (FM_NUM_APP_KEYPAD_TOGGLE) |
| CTRL X | Execute (FM_EXECUTE) |

B.4 Function key layouts

Function key layouts for specific terminal types are provided in the machine specific *MIMER Release Notes for Unix*.