OS/2 Objective C Libraries

Version 0.8

# Installation Manual

February 1996

Thomas Baier
baier@ci.tuwien.ac.at

**Abstract**

This document describes the software requirements for using this library and the installation procedure. Everytime when you install a new version of the package, you should consult this manual to find out, if anything of importance in the installation procedure has changed. Also check this document, if you encounter any problems after installing the class libraries.

If you are searching for specific information concerning

- *Installation* ··· you're reading the right manual.
- *Basics of Application development* ··· read the Tutorial. There you can find a gentle introduction into using this library package for developing OS/2 PM applications.
- *Classes and Methods provided by the library* ··· You can find special information about the provided classes and methods in the Reference Manual.
- *References on the graphical development tools (database editor, interface editor, project editor)* ··· Read the appropriate sections in the Application Programming Tools Manual.
- *Literature* ··· Look in the Literature section in the Tutorial.

# Chapter 1

# Introduction

Programming OS/2 PM applications is mostly done using the programming language C. Because the OS/2 application programming interface (*API*) is in most parts object oriented, more and more programmers choose an object oriented programming language for their purposes. The most used object oriented programming language today is *C++*.

Because of the mostly static binding and it's nearly completely missing run-time system many people are searching for easy-to use alternatives to C++. One of the most popular alternatives in object oriented programming to C++ is *Smalltalk*. Due to it's features, such as dynamic binding, messaging,... it is better suited for developing complex applications using a graphical user interface with PM.

There's another object oriented programming language, which is as easy to learn as pure C (because it's not much more than C itself), but supports dynamic binding just alike Smalltalk. This language is *Objective C*.

Objective C only adds some few new features to its "father" C, so it is an easy to learn language for C programmers.

Another advantage of Objective C is that an Objective C compiler is part of GCC, the GNU C compiler. All two ports of GCC, the EMX port, and the native port called GCC/2 support this language.

So—get it and start developing native OS/2 32bit programs using Objective C.

Additionally a simple database class library is provided to enable the programmer to access DBase III data files.

This manual describes the requirements for this library package and the installation procedure.

# Chapter 2

# System Requirements

I assume, you have EMX/GCC (emx 0.9b, gcc 2.7.2 or higher) installed on your system. To install the library you need a disk drive formatted using HPFS.

To simplify the task of designing the user interface for your programs, some programs are provided. These programs are called the Project Editor for managing the files in your projects, Interface Editor to design your program's user interface and instantiate and connect the various objects and the Database Editor to create empty DBase database files. Additionally, two programs called Console, to assist you in debugging your applications, and DBView for viewing DBase database files have been implemented.

If you are not content with the tools shipped together with this library package, you should use some other dialog editor. The dialog editor must be capable of writing `.rc` files, which can be compiled with `rc.exe`, the Resource Compiler, or ready to use `.res` files (compiled resources, binary resources). You can find a dialog editor for example in the *IBM OS/2 developer's toolkit*. I also tried to use the *Guidelines* development tool to create `.rc` files, after manually patching the generated files, even this can be used.

# Chapter 3

# Installation

To install the class library you have to unpack the compressed archive files `vd08bin.zip`[1] containing the applications, the libraries and the header files, `vd08doc1.zip` and `vd08doc2.zip` storing the available documentation in both OS/2 INF format usable to be read using "`view.exe`" and in postscript form for printing. `vd08smp.zip` at last contains the sample programs.

Change the current working directory to the root directory of the HPFS drive where you want to install the library to and type

```
unzip vd08bin
unzip vd08doc1
unzip vd08doc2
unzip vd08smp
```

This will create some directories:

- `\usr\bin` contains a single REXX script used to create objects on your desktop for the various programs and documentation files.
- `\usr\bin\PM` is the main binary directory for the applications. There you can find the development tools and all files associated with them (e.g. user interface definition files).
- `\usr\doc` contains the documentation for this libraries in postscript format.
- `\usr\include\objc` contains patched versions of "`os2.h`". This file had to be patched to work with Objective C. The original files were taken from the emx 0.9b distribution.
- `\usr\include\pm` contains the include files used for this class library.
- `\usr\include\util` contains the include files used for the utility library.
- `\usr\include\db` contains the include files for a simple database library used in one sample program.
- `\usr\lib` contains the class library for PM programming ("`objcpm.a`"), the simple database library ("`objcdb.a`"). and a library file containing some classes of general usability in the file "`objcutil.a`"
- `\usr\lib\book` contains the documentation for this libraries in OS/2 "`.inf`" format.
- `\usr\lib\icons` contains the icons used for the various folder objects.

[1] use Info ZIP 5.0 or newer

- \usr\lib\site-lisp just contains a single file named "objective-C-mode.el". This is a major mode for editing Objective C programs using *GNU Emacs*.

- \usr\samples contains the source code for the sample programs.

After installing the packages you should create objects on your desktop for the various applications and the documentation files. This is accomplished by starting the REXX script "makeobj.cmd" located in \usr\bin. If started without a parameter, the development package is assumed to be installed on drive "c:". If you have installed the package on another hard drive, you must pass the drive letter to "makeobj.cmd" when starting it.

Type

```
makeobj d
```

if you have installed the various packages on the disk drive "d:".

Finally, start a text editor program as e.g. GNU Emacs or the system editor E and edit your system's main configuration file "config.sys" located in the root directory of your boot drive.

Add "<d:>\usr\bin" and "<d:>\usr\bin\PM" to the line setting the PATH variable. Then add "<d:>/usr/include" to the variable OBJC_INCLUDE_PATH to let the Objective C compiler gcc, or in more detail, the preprocessor cpp and the compiler cc1obj, find the libraries' include files. At last you should put "<d:>/usr/lib" in front of the declaration of LIBRARY_PATH for the linker program to find the library files. To be able to view the online documentation it is recommended to add "<d:>\usr\lib\book" to the BOOKSHELF statement, too. To use the resource compiler supplied with the operating system you also have to set the environment variable INCLUDE. E.g. set it to "c:\emx\include;c:\usr\include". If you do not do this, the resource compiler probably will not be able to find the include files.

The <d:> in the paragraph above should be substituted by the drive letter including a colon where you installed the library package.

The following example will assume the library package and the applications to be installed on drive "c:". Do not just replace the appropriate lines in your "config.sys" with the ones shown below!

```
SET PATH=c:\usr\bin;c:\usr\bin\PM
SET BOOKSHELF=c:\usr\lib\book
SET INCLUDE=c:\emx\include;c:\usr\include
SET OBJC_INCLUDE_PATH=c:/usr/include
SET LIBRARY_PATH=c:/usr/lib
```

Using backslashes (\) or slashes (/) in the appropriate places is necessary, otherwise you will not be able to use the development package.

To make use of the features of the programs you will also have to install a *make* program. Project Editor expects to find a *Unix-compatible make* program called "make.exe" somewhere in your search path. It is recommended to install *GNU make* which can be found on hobbes.nmsu.edu. The *make program* must support the -C switch for specifying the compilation directory.

If the make subprocess exits with an error saying that there are too many open files in your environment and you are using a program depending on the *emx* libraries, you should increase the global file limit by setting the corresponding flag in the EMXOPT variable. To allow a maximum of 255 open files at once, put the line

```
SET EMXOPT=-h255
```

into your "`config.sys`". *make* will not be able to find the project directory if you set a *default drive* to be used for absolute pathnames starting with a slash. Be sure not to set the `-r*` option for the `EMXOPT` variable.

To support editing Objective C programs using the text editor *GNU Emacs*, a slightly modified major mode for Objective C is supplied in `\usr\lib\site-lisp`. To use this mode, add "`<d:>/usr/lib/site-lisp`" to the environment variable `EMACSLOADPATH` in your "`config.sys`" file. Then edit the editor's configuration file—either the global configuration file "`site-start.el`" or your local configuration file "`.emacs`"—and add

```
(load-library "objective-C-mode")
```

to automatically load this major mode, add

```
("\\.h$" . objective-C-mode)
("\\.m$" . objective-C-mode)
```

to the list of known file extensions and the modes these files should be edited in. The complete settings could e.g. be

```
;; Settings for some file types
(setq auto-mode-alist
      (mapcar 'purecopy '(("\\.c$" . c-mode)
                          ("\\.a$" . c-mode)
                          ("\\.h$" . objective-C-mode)
                          ("\\.m$" . objective-C-mode)
                          ("\\.tex$" . TeX-mode)
                          ("\\.texi$" . texinfo-mode)
                          ("\\.bib$" . bibtex-mode)
                          ("\\.el$" . emacs-lisp-mode)
                          ("\\.cc$" . c++-mode))))
```

Then finish configuring *GNU Emacs* by setting the global variable `user-full-name` to your name, e.g. enter

```
(setq user-full-name "John Doe")
```

This variable will be used in the automatically generated interface and implementation file headers.

To get more information on this mode, see the menu bar (a pull down menu called *Objective-C* will been installed as soon as the major mode is activated) and see the mode's description (`M-x describe-mode` will do fine).

It is recommended to compile the lisp source code by using `M-x byte-compile` in *GNU Emacs* to produce a byte-compiled "`.elc`" file which will load a bit faster.

## 3.1 Printing the Documentation

The documentation consists of four manuals at the moment. In `\usr\doc` you can find postscript versions ready to print on a postscript printer or using a postscript interpreter, e.g. Ghostscript.

- The tutorial document can be found in the file "`tutorial.ps`". It consists of 101 pages. You should at leas print out this manual. The online version might not be suited for your first steps using the libraries and the visual development tools.
- The reference manual stored in "`referenc.ps`" consists of 183 pages. Printing this manual is recommended, too.

- The user's guide to the visual development tools consists of exactly 50 pages and can be found in the file "`tools.ps`". You should at least have a look at the online version of this document.
- The installation manual, this one, should be printed. It only consists of about 15 pages and contains all information necessary for installing the package.

Printing postscript files can either be done using a postscript printer or by using a postscript interpreter. For OS/2 a version of *Ghostscript* accompanied by a simple, yet powerful, graphical user interface for viewing and printing postscript documents is available.

All documents were formatted for use with A4 paper and are suitable for two-sided printing. To accomplish two-sided printing by just printing the ood pages first, and then printing the even pages on back of the odd ones, split the postscript files using some utilities designed for this special needs. On `hobbes.nmsu.edu` you can find a port of the `psutils` capable of doing this.

The online versions of the documents were generated semi-automatically, there are no links between the topics and references are not resolved. Primarily the documents were intended to be printed. Always keep this in mind when you seem not to be able to find some topic of interest or just complain how ill-organized the documents are for online use.

# Chapter 4

# Library License

This libraries are distributed as Shareware. To become a registered user fill in the registration form in the file `register.txt` and send it to me (the address can be found in `register.txt`). You will be sent the registered versions of the package as soon as I receive the registration form (and of course after I checked with my bank that the registration fee has arrived).

After registration you are automatically registered for all following versions of the library until the major version number increases. That means by registering this version of the library together with the PM class library (PM library: version 0.8; DB library: version 0.6) you are automatically registered for all future versions of the PM and DB libraries including version 1.0.

Starting at version 1.1 of the PM or DB library you will have to register newly at a special update price.

Support the Shareware distribution concept and register if you like this library and want to use it in your own applications. Future Shareware releases of this library depend heavily on the will of users to register. So, if no one registers this library, surely no further effort will be made in adding functionality to the libraries.

As a registered user you are allowed to write applications using these two libraries and distribute them at whatever price you think of.

Before registering you are allowed to test this library package as much as you like for a trial period of 30 days after first installing this package. You are not allowed to sell any of the applications written using this package if you have not registered it.

If you continue using the library package after the trial period of 30 days and don't register, that's an act of software-piracy. May your bad conscious haunt you till the end of your days ;-)

Think of the cheap pricing for this powerful library package and register. Future releases will provide better support for Presentation Manager programming and fast database access. Supporting various database engines is also being thought of. But future Shareware-releases of this software heavily depend on the number of registrations made.

## 4.1 Distribution

This program is Shareware. Feel free to distribute the whole and unmodified package to anyone. You are not allowed to change any of the files part of the package before distributing, you only are allowed to distribute the package as a whole, including all files you received with it. Distribution of the registered version is *not* allowed. Distributing the registered version is an act of software-piracy. You probably don't want to do this `;-)`.

You are allowed to charge a small amount of money for the physical act of transferring the library. This amount of money must not exceed twice the cost of the storage medium. So, if you for example use floppy disks at a price of 10 ATS[1] each, you are allowed to charge at most 20 ATS for copying the disk. That makes a total of 30 ATS (including packaging).

If you are distributing this packages as part of a collection on a compact disk (CD), send me a copy of the disk along with pricing information.

If you don't like these distribution restrictions, simply don't distribute the program.

It's a shame to see some vendors "selling" Public Domain or Shareware programs at a price of 80 ATS per disk (3,5" HD disks are sold at a price between 5 and 10 ATS). Especially those vendors are not allowed to distribute the library package at their normal copying costs. So, change your pricing policy, or just don't distribute this library package.

If you're not sure, whether you are allowed to distribute the package, contact me. Any vendors who want to distribute registered versions of the library should do the same.

## 4.2 Warranty, Bug Reports, Support

Well, as you might have thought, there's ABSOLUTELY NO WARRANTY for this library package.

If you find any bugs in the library or want me to make improvements, drop a short E-Mail message to me at `baier@ci.tuwien.ac.at`.

If you are a registered user of the application you will be notified via E-Mail (Internet)—if possible—when new versions of the library are released. If you have any questions concerning the use of the library, working around some special problems,...send me an E-Mail message, I'll try to do my best and answer your question.

Registered users will be sent a version of the programs and libraries not limited in any way. If it is not possible to send you the programs (and the updates of course) via E-Mail, enclose a self-addressed envelope and a three disks (3,5" HD) with your registration form. Do not forget to include return postage or international postage reply coupons if you are outside Austria!

[1] ATS is *Austrian Schillings*

# Appendix A

# Future of this Development Kit

The current status of the applications and libraries shipped together with this development kit will allow you to create Presentation Manager applications mostly by just using the mouse to create windows connect objects and create custom classes.

What is still missing in the libraries is support for the “exotic” PM window classes as e.g. slider controls. I thought it more necessary to concentrate on writing some development tools assisting the application developer in everyday routine work and to provide a solid basis for application development by graphically creating classes, objects and connecting the various user interface and non-user interface objects.

For the near future an interim release of the package can be expected supporting support for index files for the database library. Although this is being worked on currently, I just got a bit disctracted by writing my diploma thesis and by finishing the documentation for this release.

In addition the classes of the database library will be supported by the visual development tools simplifying access to DBase database files a lot.

The Project Editor does not interact with the Interface Editor to my satisfaction. It should be made possible to automatically update the data in the different applications whenever another application changes something of relevance; at least the Interface Editor should automatically be notified when the Project Editor changes the class definition concerning some interface files that are currently worked on.

The database library itself should be extended to let your programs use the built-in support for DBase files as well as to access some SQL database engine, e.g. DB/2, using the same classes and methods. If you want to use a database server together with the libraries, just let me know and I'd be glad to implement support—if you can provide enough documentation and the server itself for me to test the interface.

The Interface Editor itself should be extendable by the end-user. As the simplest way to do this would be using dynamic linking using *DLL*s I'm trying to build DLL-versions of the Objective C runtime library and this package's libraries. Does anybody have an idea, why `malloc()` seems to fail if converting the runtime library “`objc.a`” to a dynamic link library?

The Interface Editor itself will be extended to support more features, e.g. creating various menus using some drag & drop actions and connecting menu items with action methods. Additionally most other connections not possible at the moment—think of simply setting the pages of a notebook control by connecting it to the various dialog windows to be displayed—will be supported.

So, once again. If you like this development kit and would like to see it grow up in the next months, support developing the package by registering.