Modula-2 System for Z80 CP/M

by

Hochstrasser Computing AG
Leonhardshalde 21
CH-8001 Zuerich
Switzerland

Manual Release 3-28-85/pwh

TABLE OF CONTENTS

STARTUP GUIDE

Chapter 6. About This Manual

Section 1. Audience

This manual describes the Modula-2 implementation of Hochstrasser Computing AG for CP/M systems using a Z80 processor. It is neither a reference manual nor a course about programming in Modula-2.

From the user's part, basic CP/M familiarity is assumed. Should you be a newcomer to programming, it is indispensable for you to buy and thoroughly study a text book about Modula-2; also, the lecture of books dealing with programming in general is recommended. For Modula-2 text books, please consult the bibliography that is contained in this manual.

When learning Modula-2, familiarity with Pascal is useful, although not necessary. The error messages of the Modula-2 Compiler are in most cases more to the point than the ones a Pascal Compiler generates, thus, it can be even simpler to learn Modula-2 than it is to learn Pascal.

Section 2. Marking and Typography

Throughout the manual, several types of enhancements are used to mark some specific sections.

Enhancement Type		Meaning
boldfaced text	-	critical actions
	-	new notions
	-	document references
underscored text	-	user input in hands-on examples.
bold, underscored text	-	most important sections in boldfaced text.

To highlight important rules, they are set between horizontal lines.

Apart from these conventions, two other kinds of marking are used:

NOTE - indented paragraphs starting with the word **NOTE** are used to mark paragraphs containing material being of special importance to the current topic of the text.

WARNING - Indented paragraphs beginning with the word **WARNING** contain information about dangers and often made mistakes related to the current topic.

Another often used abbreviation is <CR>. This means that the RETURN, ENTER, or NEWLINE key of the terminal is hit to announce the end of an input line to either the operating system or one of the programs.

Section 3. Organization

The manual is divided into different parts, each of them describing one particular topic.

Each of these parts begins with an overview describing the material covered in it in catchwords.

When you're just starting to use the system, it is recommended to search a specific topic by browsing through these catchwords to find the desired topic.

As an example, the catchwords of all of the manual's main parts are presented here in tabular form.

The main parts are:

Part	**Topics covered**
Startup Guide	- Manual usage (you're in it right now) - System description - licensing, updates, liabilities - distribution disk contents - backup and installation procedures - sample compile and run procedure
Introduction to **Modula-2**	- Modula-2 Description - Modula-2 for Pascal Programmers
Implementation **Guide**	- System Parts, Program description - How to compile and link - System Restrictions - Standard Library description - Utility Library description
Advanced Programming **Guide**	- Chaining between programs - Using shared data and heap - Assembler Interfacing - How to achieve better efficiency
Appendices	- Relocatible Format Description - Reserved Words - Standard Identifiers - ASCII Character Set (Hex/Dec/Oct) - Bibliograpy - Error Messages - Index

It is recommended to read the **Startup Guide** from cover to cover first.

WARNING - DO NOT UNSEAL THE DISKS BEFORE HAVING READ THE SYSTEM REQUIREMENTS, SYSTEM FEATURES AND THE LICENSING SECTIONS.

You can return the system and get your money back only as long as the seal isn't broken.

If you have never programmed in Modula-2 before, it is suggested to buy a good text book on it. A bibliography can be found in the appendices. There may be new good books around; choose one that appeals to you. In all cases, read the **Introduction to Modula-2** carefully. After readying the system for use, have a look at the sample programs and at the library module sources.

A **must** for everybody is the **Implementation Guide.** It contains detailed descriptions of the programs belonging to the System, as well as the included libraries.

The **Advanced Programming Guide** concentrates on more elaborate programming techniques that make the code generator produce more efficient code, as well as assembly language interfacing, operating system service usage, etc.

Chapter 7. System Description

Section 1. System Requirements

To be able to run the Modula-2 System on your computer, it has to have the following characteristics:

- Z80 or official instruction set compatible processor (i.e. NEC 780, NSC 800. Sorry, no 8080, 8085).
- **At least a 52k CP/M system.** We only tested it under CP/M 2.2, but there seems to be no obstacle against earlier or later versions. The larger your system, the larger the programs the compiler can translate.
- The compiler itself uses about 170k Bytes of disk space; though it is desirable to have **at least 2 drives holding 300kBytes each** to work comfortably with the system.

Section 2. System Features

The Modula-2 System for the Z80 **generates fast, ROM-able, reentrant Z80 native code. Assembly language integration** is supported, as well as assembly language compiler output. The System also sports an **auto-search mechanism** in the compiler as well as the linker.

However, some minor omissions are present in this package: There are no priorities and therefore no monitors set up by the compiler. This omission was made because of the Z80's "flexible" interrupt structure: There are nearby as many different interrupt schemes as Z80 based computer brands. Version Control is currently disabled.

Chapter 8. Licensing, Updates, Liabilities and Support

The **license agreement** (see separate sheet) allows for single CPU usage of the Modula-2 System for Z80 CP/M as well as for marketing your own programs, given you give credit to us and the Modula-2 System.

If you fill in, sign and return the license agreement to us, you are eligible for discounted updates to later versions of our product, provided you respected the license agreement's terms.

We feel that this license agreement is quite fair and wish to thank Leor Zolman of BDS, Inc. for allowing us to merely copy this text from his product, the BDS C System.

Please note that the **standard disclaimer applies** to this software. This means that we can't take any liabilities that you can accomplish a specific task using this Modula-2 System. However, because all the programs of the system as well as some commercial applications are written using this system itself, it has been proven that it really is usable and working.

Because we are a small company and have no branch offices, we cannot offer a dedicated telephone support line to our customers from overseas. Naturally, you can contact us with letters and even phone calls. Please note the time difference between Europe and other continents in the latter case. Switzerland, in particular, is in the GMT + 1 zone.

We are eager to get any and all bugs you may encounter out of our products as fast as possible, and count on your help in this respect.

Chapter 9. **Getting Started**

This section gives an overview of the components of the Modula-2 System, explains how to adapt it to your computer and features a sample compilation. Therefore, please read this section carefully.

Section 1. **Distribution Disks**

The distribution diskettes contain your Modula-2 System.

WARNING - Do not use these disks as your work disks. Please make at least one backup prior to any work with the system. That's in your very best interest.

The following files can be found on the distribution disks:

Group	**File**	**Contents**
Compiler	MC.COM	Syntax check
	MCP2.COM	Imports & Initialisation
	MCP2A.COM	Declaration Analysis
	MCP3.COM	Body Analysis; Type Check
	MCP3A.COM	Symbol File Generation
	MCP4.COM	Code Generation, MRL output
	MCP4MAC.COM	" , Assembler output
	MCL.COM	Lister
Linker	ML.COM	Pass 1
	MLP2.COM	Pass 2
Ref-Lister	MX.COM	Reference Lister

Group	File	Contents
Converter	MR.COM	REL to MRL converter
Installation	MP.COM	System Configuration Program
Library	MODLIB.MRL	Runtime Library
Utilities	CMDLIN.*	Command Line Scanner
	CONVERSI.*	Conversions (numbers <-> strings)
	CONVERTR.*	Conversions (REAL numbers <-> strings)
	INOUT.*	Standard I/O
	REALINOU.*	Standard I/O for REALs
	MOVES.*	fast moves
	OPSYS.*	operating system access
	SEQIO.*	Sequential File I/O
	FILESYS.*	Sequential File I/O
	FILES.*	Random File I/O
	TERMINAL.*	CRT I/O
	TERM1.*	CRT I/O (sample programs)
Test Programs	ACKER.MOD	Ackermann function
	ERATOS.MOD	Sieve of Erathostenes (BYTE)
	QUEENS.MOD	Eight Queens Problem

These files will be present in any case; there might be others as well.

Section 2. Making a Backup Copy

Your first step is now to **make a copy of the distribution disks.** This copy will serve as your Copy Master thereafter.

To generate a copy, follow these steps:

- prepare enough diskettes to hold the system. This means about 450 kBytes of diskette storage. Each of these diskettes should be formatted, SYSGENized and hold a copy of PIP, CP, or your favourite copy program. It is desirable to have also a text editor as WordStar, Mince, PolyVue or ED and SUBMIT on these disks.
- Copy **all files** from the distribution disks to your prepared disks. Use these as your copy masters from now on, i.e. all copies of the system to generate a new work disk will be taken from these disks.
- Perform a configuration. See the following section for this.
- make a SUBMIT-file, which permits you, to copy automatically all necessary files to make a work disk on a new disk. Such a file might look like:

```
PIP $1:=MC*.*[v] <CR>
PIP $1:=ML*.*[v] <CR>
PIP $1:=MODLIB.MRL[v] <CR>
PIP $1:=MX.COM[v] <CR>
PIP $1:=TERMINAL.*[v] <CR>
PIP $1:=SEQIO.*[v] <CR>
ERA $1:TERMINAL.MOD <CR>
ERA $1:SEQIO.MOD <CR>
```

  Let's call this file "MODULA2.SUB". Now insert your master disk containing this file into drive A: and the work disk-to-be into drive B:. The correct usage of the SUBMIT file would be:

```
A>SUBMIT MODULA2 B <CR>
```

By executing these steps, you created your master diskettes. From these, you can create all of your work disks. The components that are necessary to create a working system are: The compiler (MC*.*), the linker (ML*.*), as well as the runtime library (MODLIB.MRL). If you want to create listings without a compilation but with numbered lines and reference tables, you need also MX.COM. All other modules mentioned in the SUMBIT file can be replaced by modules developed by yourself.

WARNING - Hochstrasser Computing AG and its distributors do not provide replacements for intentionally or carelessly damaged distribution disks. Therefore, do **NOT** use the distribution disks as your master disks or - even worse - as your work disks. Keep them "off premises" in a safe or some other secure place. Make copies of them only if necessary, i.e. if the other masters are unusable!

Section 3. System Configuration

The Modula-2 System features an Auto-Search-Mechanism. This allows the compiler to search for files which cannot be found on a particular drive, on other drives. The system distinguishes between two so called **search paths:** one for Data-Files (input files) and one for its own program parts. With the aid of MP.COM, these two paths can be installed into the system.

For the installation, the following files have to be present on your disk:

MP.COM	-	The installation program
MC.COM	-	The compiler base
ML.COM	-	The linker base

Now invoke MP as follows:

NOTE - user input is -according to the typographic conventions- underlined.

```
A>MP <CR>
MODULA-2 Patcher for Z80 CP/M  Version 15-03-1985

Processing Compiler

enter data search path       (Default @AB    ): <DataPath> <CR>
enter program search path    (Default @AB    ): <ProgPath> <CR>

Processing Linker

enter data search path       (Default @AB    ): <DataPath> <CR>
enter program search path    (Default @AB    ): <ProgPath> <CR>

Installation Complete

A>
```

DataPath indicates where the programs shall get their input files from.

ProgPath tells the programs where to search their parts. You can spread, for instance, the compiler's parts among seven drives, if you like to do so (the compiler consists of seven files). All of the programs make internal version controls, so any mismatching (old) versions are detected. Any operation is aborted immediately in this case.

<DataPath> as well as <ProgPath> both consist of one to six letters. These letters can be either "@" (the at-sign) or "A" up to and including "P". The at-sign denotes the so called **default drive**, the other letters correspond to the respective drives allowed under CP/M.

The default drive is the one whose letter is used in CP/M's prompt: If the prompt is **A>**, then the A: drive is the current default drive, if it is **F>**, then F: is the one. If you do not indicate where CP/M shall get a file or program from, it takes it from the default drive. This drive is often referred to as the **currently logged drive.**

NOTE - There is no way to direct the intermediate files of the compiler to another than the default drive.

On a standard dual disk drive system, assuming that your copy master disk is in drive A:, MP would receive the following answers:

```
Compiler Data     Search Path: @
Compiler Program  Search Path: @AB

Linker Data       Search Path: @
Linker Program    Search Path: @AB
```

This directs the programs to search their input on the default drive only, and their program parts first on the default drive and then on the A: drive, and finally on the B: drive.

If you have, say a North Star Advantage with only a 320 kBytes per disk capacity, it might be reasonable to place all of the program parts on the B: drive, including the runtime library, while your programs would be placed on drive A:. In this case, your answers change to:

```
Compiler Data     Search Path: @BA
Compiler Program  Search Path: B@

Linker Data       Search Path: @BA
Linker Program    Search Path: B@
```

So, the linker will search for the runtime library first on the default drive (usually A:) and then on B:. The 'A' in the DataPath is included to allow compilations on B: that require files from A: to be successful.

NOTE - The installation program checks for multiple usage of the same drive and flags it as an error.

Section 4. A Sample Compile and Run Session

If you aren't that much curious to see "how it runs", you can skip this section and switch to the **Introduction to Modula-2.**

On the distribution disks, you find some example programs, which are ready to compile. One of them is "ERATOS.MOD", a Sieve of Erathostenes benchmark. By following the steps explained below, you can do a sample compilation. Please do not forget **to install the compiler <u>before</u> using it.**

- On your work disk, there should be the following files:

ERATOS.MOD	- the sample program.
TERM1.MSY	- the **symbol module** of Term1.
TERM1.MRL	- the **relocatible file** of Term1.
MODLIB.MRL	- the Modula-2 runtime library.
MC*.*	- the **installed** compiler.
ML*.*	- the **installed** linker.

- Start the compilation as follows:

```
A>MC ERATOS
Modula-2 compiler for Z80 CP/M    Version 15-03-1985
Compilation of ERATOS.MOD
P1.
Imported Modules:
    TERM1    .MSY - TERM1
P2.
P3.
P4.

A>ML ERATOS
Modula-2 linker for Z80 CP/M    Version 20-03-1985
P1....................
P2....................

A>ERATOS
There were 1899 primes.
```

NOTE - The dates in the above example do not necessarily coincide with what you get from your distribution disks. Also, the number of dots output by the linker can change.

Further information about the programs, their operation and usage is covered by the **Implementation Guide.**