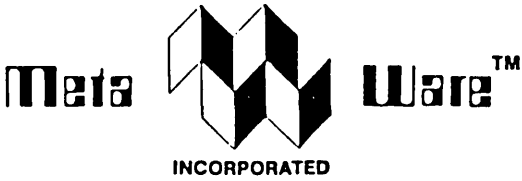

High C ™

Installation Guide

Version 1.2 Concurrent DOS

by MetaWare™ Incorporated

High C ™

Installation Guide

Version 1.3 for Concurrent DOS 286

NOTICES

The software described in this guide is licensed, *not sold*. Use of the software constitutes agreement by the user with the terms and conditions of the End-User License Agreement packaged with the software. Read the Agreement carefully. Use in violation of the Agreement or without paying the license fee is unlawful.

Every effort has been made to make this guide as accurate as possible. However, MetaWare Incorporated shall have no liability or responsibility to any person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this guide, including but not limited to any interruption of service, loss of business or anticipated profits, and all direct, indirect, and consequential damages resulting from the use of this guide and the software that it describes.

MetaWare Incorporated reserves the right to change the specifications and characteristics of the software described in this guide, from time to time, without notice to users. Users of this guide should read the file named "README" contained on the distribution media for current information as to changes in files and characteristics, and bugs discovered in the software. Like all computer software this program is susceptible to unknown and undiscovered bugs. These will be corrected as soon as reasonably possible but cannot be anticipated or eliminated entirely. Use of the software is subject to the warranty provisions contained in the License Agreement.

A. M. D. G.

Trademark Acknowledgments

The term(s)	is a trademark of
Digital Research	Digital Research, Inc.
Concurrent, CP/M	Digital Research, Inc.
Concurrent Helper	MetaWare Incorporated
High C, MetaWare	MetaWare Incorporated
Professional Pascal	MetaWare Incorporated
UNIX	AT&T Bell Laboratories, Inc.

Installation Guide
for the
High C ™ Compiler

Version 1.3 for Concurrent DOS 1.2 and Later

Table of Contents

1.0 Requirements and Assumptions

1. Hard disk space needed with

   Compiler, associated support software, and
   Small memory model library ..... 1.7 MB
   Libraries for additional memory models ..... 0.9 MB
   Utilities ..... 0.2 MB

   Allow about 230KB per library pair: co-processor and emulator versions.

2. ???KB of main memory to run the compiler in non-overlaid mode, and ???KB in overlaid mode. But note: use of the **-ansi** option requires about 15K more — the **-ansi** option turns off any and all extensions beyond the (proposed) ANSI Standard. [[[??? Fill in KB sizes above.]]]

3. You have a 1.2MB floppy disk drive designated as A:, and a hard disk drive known as C:. *All installation procedures install from A: and onto C:*. If these drive letters are different on your system, make the appropriate substitutions when typing in commands from this document, and change the contents of the supplied .BAT files, or use the Concurrent ASSIGN command to change drive letter assignments.

2.0 Installation

Due to the compiler's size, you must install it on your hard disk before using it. To install the compiler, do the following:

1. Create a directory on your hard disk where you want the compiler and associated software to reside. Make that directory your working directory. For example:

   ```
   md /highc
   cd /highc
   ```

   to create a directory at the root called "highc" and make "highc" the working directory.

2. Get your floppies ready. Put the floppy labeled #1 into the A: drive.

3. Type "a: **install**". INSTALL will prompt you to load the appropriately numbered floppies and it will extract the information from them.

3.0 A Small Demonstration

After the compiler is installed, hc.286 — the non-overlaid version of the compiler — and the Run-Time Library for the Small memory model reside on the hard disk. You can then get a small demonstration of the compiler in operation by entering:

smaldemo

If you do not have enough memory to run the compiler non-overlaid, you will have to run the INSTALL2 batch file to install the overlaid compiler (on diskette #3), or you can just copy it yourself. The overlaid compiler is named hcov.286. Replace hc.286 with the overlaid version by typing

copy A:hcov.286 C:hc.286

4.0 Changing Memory Models

Typically, you will write programs that use one memory model; the next section below tells how to install just one model. The compiler supports five memory models to make optimal use of the 8088/8086/186/188/286 processor. Refer to the **Programmer's Guide** for a description of the memory models.

To install the rest of the models (and the utility programs described in Section *Utilities* below), make sure that you still have about 1.1 MB of free hard disk space, then enter:

install2

You can see a demonstration of programs running in all the memory models by typing

alldemo

5.0 Installing an Individual Memory Model

To install the Run-Time Library of a given memory model, copy its directory from distribution diskette #2. See below for a list of the diskettes and their contents.

For example, copy the compact directory from diskette #2 to get the Library for the Compact model. If your want to replace one or more Libraries of other memory models with this Library, you should delete their directories in /highc, either before or after this step. In general, you may delete any Library directory at any time is is not needed.

The supplied batch file GETMOD automates the selection and copying of a particular memory model. Type

```
getmod memory_model_name
```

where memory_model_name is small, medium, compact, big, or large (typed in all lower case). Also, if memory_model_name is utils, GETMOD copies the utilities described next.

6.0 Utilities (Concurrent Helper™)

The Compiler and Run-Time Libraries are distributed with a set of UNIX™-like utility programs, called Concurrent Helper™, that can much enhance productivity on Concurrent. The utilities provide a host of capabilities lacking in the operating system itself, such as file archiving and searching functions. The utilities can be used to maximize performance and organization under the versatile Concurrent hierarchical directory structure.

Concurrent Helper consists of eight distinct programs:

FIND facilitates general-purpose, powerful directory maintenance and archiving. (FIND is used by the INSTALL program to load the compiler.)

TAIL prints the "tail" (last few lines) of a file.

MV renames one or more files or moves them from one directory to another.

LS lists the contents of directories in various useful formats.
CAT concatenates a sequence of files into a single file.
UNIQ removes duplicates of lines from a file.
FGREP rapidly searches a file or files for a string pattern. (FGREP is an attractive alternative to a bulky printed paper cross-reference, since it is very fast.)
WC counts the number of lines and words in one or more files.

These utilities are described in detail in the on-line documentation on the last diskette.

7.0 Compiling and Executing Programs

The **Programmer's Guide** tells you how to compile, link, and execute programs. But just to get you started quickly, here are the essential instructions:

1. Edit a program into a file "hello.c". For example,

```
main() {
   printf("Hello, world.");
   }
```

   (demos/hello.c contains a copy of this program.)

2. Run the compiler:

```
hc hello
```

   just to compile, or

```
hc hello -on list > hello.l
```

   to produce a listing and send it to file "hello.l".

3. Link the program:

```
link hello,small/hcse.186
```

4. Run the program:

```
hello
```

   You should receive a greeting.

8.0 Recommended File Organization

Once you have installed the compiler, it resides in its own directory which we will assume is called /highc. However, you will probably not want to do program development from this directory. A better idea is to leave the /highc directory alone and do your work elsewhere.

To do this, you must arrange matters so that the compiler, and preferably Concurrent Helper, are easily executable from your program development directories, and so that the Run-Time Library(-ies) you link to are easily available.

To make the compiler and utilities executable from any directory, /highc/hc.286 and the .286 utility files must be visible on your execution path. We recommend that you construct a /bin directory (following UNIX conventions, for "binary executables"), if you have not already, and place all common executables there.

Instruct Concurrent to search the /bin directory for hc.286 by adding "c:/bin" to the "path". For information on "path", see the Digital Research Concurrent DOS User's Guide manual section on Concurrent DOS Commands.

Assuming that you run INSTALL2, or use "**getmod** utils" for the utilities only, as presented above, you must first move all the utilities into your /bin directory. Enter

```
cd /highc/utils
mv *.286 /bin
```

which moves all the utility programs into /bin. The "**mv**" command is one of the Concurrent Helper utilities.

Now move the compiler "hc.286" to /bin from /highc by entering:

```
mv /highc/hc.286 /bin
```

Any Run-Time Libraries you use may also reside in /bin. For purposes of illustration, assume that you intend to use the

Small memory model with floating-point emulation, so that the Run-Time Library you will link to is "hcse.186". Enter:

```
mv /highc/small/hcse.186 /bin
```

to move the library there.

High C comes with the standard ANSI-required ".h" header files. These files are located in the "/highc/inc" directory. In addition there are a collection of "utility packages" with suffix ".cf" whose interfaces are located in the "/highc/inc" directory. The packages provide access to Concurrent directly and supply useful utility routines. See both the **Programmer's Guide** and the contents of the files in "/highc/inc" for more information.

You will want to access the package interfaces from whatever directory you work in. One way of doing this is to use a full path name when including the package interface:

```
#include "/highc/inc/stdio.h"
main () {
   int i;
   for (i = 0; i <= 9; i++)
      printf("%d squared is %d/n",i,i*i);
   }
```

However, the text "/highc/inc/stdio.h" in your program requires program changes should you want to move the "inc" directory elsewhere. To avoid this inconvenience you can direct the compiler to search a sequence of directories when opening input files (such as a **#include**-d file above) by using the Concurrent "**define** ipath=" command ("ipath" for "input path"). For example:

```
define ipath=/highc/inc/;/other/stuff/;/etc/;
```

instructs the compiler to look first in /highc/inc, then in /other/stuff, then in /etc for input files. (For information on "**define**", see the Digital Research **Concurrent DOS User's Guide** manual section on Concurrent DOS Commands.) With this, the above program can be simplified:

```
#include "stdio.h"
main () {
   int i;
   for (i = 0; i <= 9; i++)
      printf("%d squared is %d/n", i, i*i);
   }
```

If you decide to change where stdio.h resides, you need only change the "ipath". For further information on "ipaths", see Section *Compiler Controls* in the **Programmer's Guide.**

You may want to change your AUTOEXEC.BAT file to include the "**define** ipath=" command shown here. Then the "ipath" will be defined every time the system is booted.

Alternatively, the form of **#include** in which the file is not quoted but instead enclosed in the characters < and > searches first an "<>-include" search path that you can set up at installation time using the CONFIG program; see *Configuring the Compiler* below. If you specify the value "/highc/inc/" for the <>-include search path, you can write

```
#include <stdio.h>
main () {
   int i;
   for (i = 0; i <= 9; i++)
      printf("%d squared is %d/n", i, i*i);
   }
```

to obtain the inclusion of /highc/inc/stdio.h.

If you plan to use the **-ansi** option, you *must* place the /highc/inc directory on the "ipath", since files necessary for operation in ANSI mode are stored in /highc/inc.

9.0 Distribution Diskettes Contents

The contents of the distribution disks are as detailed below. On diskette #1 you will not actually see all the files. Some are "hidden" inside a large ".tar" file and are extracted when you run the INSTALL program discussed above. However, the library directories are directly visible on the second diskette, so it is easy for you to extract the portions of a desired memory model subdirectory.

Any name listed below that ends with a / is the name of a (sub-)directory.

Diskette No.	*Contents*	*Description*
1	install.doc	The text of this document.
	readme	Last minute installation notes.
	find.286	Utility program to load the software.
	demos/	Various demonstration programs.
	bench.mks/	Common bench-mark demos.
	inc/	.h and .pf interface files for accessing standard ANSI header files and the utility packages in the Run-Time Libraries. See the **Library Reference Manual** and the **Programmer's Guide.**
	hcansi.st	ANSI scan and
	hcansi.pt	ANSI parse
	hcansip.pt	tables.
		(Needed for the **-ansi** compiler option.)
	hc.286	The compiler.
	xref.286	The cross-referencer.
	*.bat	Various batch files.
	config.286	Compiler configuration program.

	bd.286	A binary dump utility that readably displays the contents of .OBJ and .186 files. Specify "**bd** f1 f2 ..." to dump files f1, f2, ... To understand its output you must know Intel OMF; consult part number 121748-001, **8086 Relocatable Object Module Formats**, available from Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.
	clname.286	Utility to change Lnames in .OBJ files. Consult the README file for more information.
	lib/	
	src/	Library sources currently being distributed. See Section *Embedded Applications* in the **Programmer's Guide** for more details on the source. See also *Embedded Applications* below.
	etc/	Last-minute or unclassified software.
2	small/	Run-Time Library for the Small model.
	hcsc.186	hcse.186 ptoc.186
	c_close.obj	c_heap.obj c_scanf.obj
	c_printf.obj	cfinit.obj heap1.obj
	stkdmp.obj	hc.pro pp.pro
	model.a86	
	medium/	Run-Time Library for Medium model.
	hcmc.186	hcme.186 ptoc.186
	c_close.obj	c_heap.obj c_scanf.obj
	c_printf.obj	cfinit.obj heap1.obj
	stkdmp.obj	hc.pro pp.pro
	model.a86	
	compact/	Run-Time Library for Compact model.
	hccc.186	hcce.186 ptoc.186
	c_close.obj	c_heap.obj c_scanf.obj
	c_printf.obj	cfinit.obj heap1.obj
	stkdmp.obj	hc.pro pp.pro
	model.a86	

big/	Run-Time Library for Big model.		
	hcbc.186	hcbe.186	ptoc.186
	c_close.obj	c_heap.obj	c_scanf.obj
	c_printf.obj	cfinit.obj	heap1.obj
	stkdmp.obj	hc.pro	pp.pro
	model.a86		
large/	Run-Time Library for Large model.		
	hclc.186	hcle.186	ptoc.186
	c_close.obj	c_heap.obj	c_scanf.obj
	c_printf.obj	cfinit.obj	heap1.obj
	stkdmp.obj	hc.pro	pp.pro
	model.a86		

3	utils/	MetaWare Concurrent Helper utilities.
	doc/	Documentation for the utilities.
	hcov.286	Overlaid version of hc.286. Uses much less memory but takes longer to run due to operating system loader overhead.

The contents of each of the five model subdirectories are similar. For example, the contents of the small/ subdirectory are:

hcsc.186	hcse.186	ptoc.186
c_close.obj	c_heap.obj	c_scanf.obj
c_printf.obj	cfinit.obj	heap1.obj
stkdmp.obj	hc.pro	pp.pro
model.a86		

The first file is the Run-Time Library ("hc") for the Small model ("s") that assumes an 8087 co-processor ("c"). The second does not assume an 8087 ("e" = emulator), but will use one if present.

ptoc.186 and cfinit.obj are used when linking with programs written in MetaWare's Professional Pascal language. stkdmp.obj and heap1.obj are used to obtain certain post-mortem traces. c_heap.obj is used to (almost) completely eliminate the heap manager if desired. c_close .obj, c_scanf .obj, and c_printf.obj similarly reduce memory requirements. See the **Programmer's Guide**, Section *Linking a Com-*

piled Program, for more details. The source of c_heap.obj is lib/src/c_heap.c.

For each memory model M, the hc.pro file is a compiler "profile" for use when compiling programs in M. It contains a pragma to set the memory model to M. See the **Programmer's Guide** on the Memory_model pragma and the profile. The pp.pro file is a similar profile, but for MetaWare's Professional Pascal language. Use it when recompiling for model M supplied Run-Time Library source in lib/src; you will need this only for doing embedded applications programming — and you will need the Professional Pascal compiler for Concurrent.

File "model.a86" is used for assembling supplied Run-Time Library source; see *Embedded Applications* below.

The disk containing the MetaWare Concurrent Helper utility programs contains all their documentation. There is no supplied hard-copy documentation for them.

10.0 Configuring the Compiler

Various compiler defaults are modifiable through directives either specified on the compiler invocation line, such as "**hc** program **-off** Warn" or in the source code of a program, such as "**pragma** Off(Warn);".

The CONFIG program provides a way to change some of these defaults within the compiler itself, so that the defaults hold for every program compiled. The CONFIG program actually modifies the compiler executable file. The changed defaults hold until changed again by the CONFIG program.

Some of the configurables are: run-time checks; the number of tree pages in memory during compilation; the memory model to generate code for; whether to generate code for a 186/286 processor; the assumed suffix of source files when none is supplied with compiler invocation; etc.

The CONFIG program is self-documenting. It tells you what defaults are configurable and guides you through changing

them. For details on what option, pragma, and toggle defaults are configurable, consult the **Programmer's Guide.**

The configurable options are generally those that you set up once and then forget. In particular, you will probably want to set the **-tpages** value to one appropriate for your memory configuration.

[[[Appropriate suggestions for tpages?]]] The more **-tpages**, the less paging of the compiler's internal tree to disk, and the faster the compilation.

(Note that most "memory-mapped disk drives", also called "mdrives" or "RAM disks", consume available memory; if you have such a drive, be sure to adjust your computation of available memory. If you have never heard of "mdrives", you probably do not have one and can ignore this parenthetical remark.)

11.0 Embedded Applications and Supplied Source Files

For special applications environments, the source of some of the Run-Time Library has been provided. Read the **Programmer's Guide** Section *Embedded Applications* for more information as to the purpose and use of this source.

You can make any changes to the library source that you like, but note that all source in `lib/src` is copyrighted by MetaWare. You may use the source as long as only the object code of it leaves your premises. In essence, you have the same rights to the object code from this source as you do to the Run-Time Library object; you have the same rights and obligations with respect to the source as you do the compiler object.

The MetaWare copyright notice must be preserved and be present in any linked .286 files that use any portion of the MetaWare Libraries. The `init.a86` source has a copyright notice in the data definition for the initial run-time stack that satisfies this requirement if it is preserved. If you replace

init.a86 entirely you must include such a copyright notice if you use any portion of the library.

Assembly source is written in Digital Research's "rasm" assembly language, not Intel assembly language. To assemble it with a specific model in mind (Small, Medium, Big, etc.), the macro file "model.a86", specific to each model, must be included. It sets the parameters for the model. Therefore there are five "model.a86" sources. Each can be found in the directory by the same name as the model.

Thus, for example, to assemble init.a86 for each of the five models, you would need the following files:

```
 lib/src/init.a86
   small/model.a86
compact/model.a86
  medium/model.a86
     big/model.a86
   large/model.a86
```

Other library routines are written in a combination of MetaWare's High C and Professional Pascal; to recompile them for a particular memory model, use the pp.pro (for Pascal) or hc.pro (for C) file contained in that model's subdirectory to set the model. Of course, you may need a copy of both compilers for Concurrent, depending upon which routines you are recompiling.

Alternatively you may choose to replace them altogether with your own routines. At least by looking at the source you can see how the implementation was done and what the programs must do for embedded applications.

12.0 Solutions to Potential Problems

Here are a few problems that you might encounter, and what to do with them.

1. The compiler complains that it is "Out of memory.".

   If you have limited main memory on your machine, you must set the number of tree pages down to a small number, such as 50 or even less. For example:

   ```
   hc program.p -tpages 50
   ```

   You can also "configure" the compiler to have 50 tree pages and to assume limited memory; see *Configuring the Compiler* above.

   Note that running with the **-ansi** option requires about 15K more memory.

2. You have successfully compiled a program, but the linker complains that one of the symbols SMALL?, MEDIUM?, COMPACT?, BIG?, or LARGE? is undefined.

   This means that you are linking with the wrong library. Programs compiled for a particular memory model must be linked with the library for that model. See the **Programmer's Guide** for more information on memory models.

3. You have successfully compiled a program, but the linker complains that the symbol _mw87_used is undefined.

   This means you have object files that contain 8087 or 80287 floating-point instructions, but you are linking with an emulation library. Programs compiled into -87 floating-point instructions must be linked with the *co-processor* library for that model. See the **Programmer's Guide** for more information on libraries.

4. You have typed

   ```
   hc
   ```

to start up the compiler and it has printed a banner but is doing nothing else.

It is waiting for the program to be entered from the terminal. When no parameters are given to the compiler, it assumes that the input program is coming from the standard input, which in this case is the console. You can type the program in directly, terminated by a control-Z to signify end-of-file (and end-of-program).

Acknowledgments

The authors of these manuals and designers of Professional Pascal would like to thank Victor Technologies Incorporated for the initial impetus to actually implement our own Pascal. We especially thank Marq Elliott for his enthusiastic support, along with Tony Jeans and Bobby Taylor.

We thank three enthusiastic people at "beta" test sites for their valuable feedback on the manuals, the language, and the compiler. They are John Ciccarelli of Zoran Corporation, Wes Embry of Daisy Systems Corporation, and Reed Kotler of General Transformation Corporation.

Thanks also to Professor Bill McKeeman of the Wang Institute of Graduate Studies for his valuable feedback and to one of our marketers, the people at Microtec Research Incorporated, especially Craig Franklin, for their insights into the needs of typical users, for many suggestions for technical improvements, and for the evaluation of the language which they supported by Gene Powers. And thanks for your insights, Gene.

Most of all we acknowledge that we are not self-made, but God-made. And we thank God for building into us the talents that made it possible for us to create Professional Pascal.

Ad majorem Dei gloriam (A.M.D.G.).

This ends the

MetaWare™ High C ™

Installation Guide