METAFONT for the *AMIGA*

Documentation of version 2.71

Andreas Scherer
Abt-Wolf-Straße 17
96215 Lichtenfels
Germany

March 27, 1994

Abstract

This document describes a new implementation of METAFONT 2.71 for the Commodore Amiga. METAFONT is the name of a computer font generation system invented by Prof. Donald E. Knuth of Stanford University. It reads text files prepared with 'programs' that specify the outline of characters in fonts of type, graphical symbols, or other elements for printing high quality documents. Typesetting systems like TEX make use of the font metric information produced by METAFONT and printer drivers or screen previewers make use of the pixel information as to where to put the 'ink.' The usage of the associated programs is described to some detail, but no introduction to the METAFONT language itself is given. Refer to the bibliography (section 6 of this text) for detailed information about METAFONT programming.

Contents

1 Copyright

The METAFONT program and the associated utilities are copyright © 1984 by Donald E. Knuth. All rights are reserved. The complete WEB source code is published as [5].

This new implementation of METAFONT 2.71 for the Commodore Amiga is declared to be FREEWARE. All rights are reserved. You may use, copy, and distribute the package in complete archived or installed form and for a nominal fee for media and shipping only.

Although considerable effort has been expended to make the METAFONT program correct and reliable, no warranty is implied; the author and the implementor disclaim any obligation or liability for damages, including but not limited to special, indirect, or consequential damages arising out of or in connection with the use or performance of this software.

This work has been a "labor of love" and both the author and the implementor hope that users enjoy it.

Please send comments, suggestions, bug reports, etc., to Andreas Scherer, Abt-Wolf-Straße 17, 96215 Lichtenfels, Germany. A "developer's kit" containing the complete source code with all modifications and enhancements is available at this address. (See section 8 for details.)

TEX is a trademark of the American Mathematical Society.
METAFONT is a trademark of Addison-Wesley Publishing Company.
Commodore is a registered trademark of Commodore Electronics Limited.
Amiga, AmigaDOS, Amiga Kickstart, and Amiga Workbench are registered trademarks of Commodore-Amiga Incorporated.
SAS and SAS/C are registered trademarks of SAS Institute Incorporated.

2 System requirements

The minimum configuration tested with this implementation of METAFONT was a Commodore Amiga 2000 running the standard 68000 Motorola processor and 2 MB of ChipRAM, although a total of only 1 MB were actually used with the default configuration. Memory requirement may increase depending on the font to be created and the settings in the METAFONT configuration. (See section 4.2.) The operating system was Kickstart 37.175 and Workbench 38.36. All programs included in this package are capable of running under version 1.3 of the operating system. Only the online display of METAFONT requires version 2, because of the use of some advanced graphics routines, but this feature is protected when used with version 1.3.

There are special versions of the IniMF and VirMF programs provided for advanced hardware configurations like the Amiga 3000 and Amiga 4000 or Amigas

with accelerator boards. These two programs rely on features of the 68020 Motorola processor not present in the 68000 processor. They will run approximately twenty to thirty percent faster, but they will crash on non-accelerated Amigas! All other programs that come with this package are compiled for use on any standard Amiga configuration.

METAFONT is a large system intended for large tasks, so this is no greasy kids' stuff! The creation of `cmr10` for the 300 dpi DeskJet printer driver of PasTEX can be as quick as 48 seconds on a 50 MHz 68030 accelerator board with a coprocessor and sufficient 32 bit FastRAM but as slow as 10 minutes and 5 seconds on a standard 7 MHz Amiga with slow 16 bit RAM, although the programs are capable of running on both configurations.

3 Compatibility

This is METAFONT 2.71 for the Commodore Amiga. It is fully compatible with the original implementation by Donald E. Knuth and has passed the TRAP test as of January 25, 1992 gracefully. Only permissible differences as described in `trapman.tex` were encountered. `diff` files documenting the latest run of the TRAP test will be made accessible by the implementor on request. (See section 8 for details.)

Also included are GFtoDVI version 3.0 (October, 1989), GFtoPK version 2.3 (July, 1990), GFtype version 3.1 (March, 1991), MFT version 2 (October, 1989), PKtoGF version 1.1 (October, 1990), and PKtype version 2.3 (November, 1989).

4 Installation

4.1 Extracting the archive

This "METAFONT user's kit" is a self-contained system; everything you need for the creation of beautiful fonts for TEX is here. This subsection describes the various subdirectories and their contents. Normally this package should come in the form of an `LHA` archive. It will contain a working installation of all programs and scripts, except that some environmental settings have to be installed by hand. Sorry, there is no 'Installer' script, but installation is really very easy. First you have to extract the files by the command

```
> LHa x MetaFont[.lha] <path or device for installation>
```

(You should have the `LHa` program to do this. If you don't, look for Fish Library Disk 715 for the current version.[1]) A root directory `MetaFont` will be created and will contain the complete archive in convenient subdirectories described below. You should "`assign`" the name "`MF:`" to this root directory. Less than 3 MB of free disk space are needed to install the complete METAFONT system.

[1] But don't send any money to Stefan Boberg, because you won't get what you want!

`MetaFont/bases`: This directory is the home of the 'base' files that METAFONT reads at high speed at the beginning of a run. Here you will find the file `plain.mf` (version 2.71) and some startup files for 'plain' and 'cm.' New base files should be created or copied herein. The environment variable `MFBASES` should contain an entry pointing to this directory in order that METAFONT can find them. (See section 4.3 for details about the environment variables recognized by METAFONT.)

`MetaFont/bin`: The most important part of this package are the binary files of the programs for METAFONT and the associated utilities. You should tell the operating system to look for the programs here by adding the installation path to the `PATH` environment variable. A more detailed description about running the programs will be given in the next section 5.

`MetaFont/config`: The present implementation of METAFONT is capable of resetting the memory configuration on startup in case the inner settings are too meager for a given task and METAFONT aborts its run with the error message `METAFONT capacity exceeded, sorry...` The configuration setup file `mfmemory.config` contains lines of the form

```
set memmax              65530     % default 30000
set screenheight          200     % default  1024
set <internal size> <user value> % default <value>
```

You may change the settings (in decent ranges!) or delete some lines from the file, thus resetting to internal values (given in the 'default' column). The environment variable `MFCONFIG` should contain an entry that points to this directory. (See section 4.3 for details about the environment variables recognized by METAFONT and section 4.2 for details about the memory configuration.) The file `modes` contains a list of settings appropriate for the ready-to-work base file provided with this user's kit[2]. Several printers and the standard previewer are supported. You may change things here according to your personal configuration.

`MetaFont/doc`: At least this directory you have already visited if you are reading this file. Here you find the documentation for METAFONT 2.71 in both German and English. The LaTeX source files must be present, the `dvi`-output may have been deleted due to space limitations, but you can recreate it with a running TeX installation; no special macros or unusual fonts are necessary for this file.

`MetaFont/inputs`: This directory is for METAFONT programmers and system administrators only. It contains text files necessary for the creation of

[2]These are **not** compatible with the PasTeX settings from `pastex-modes.mf` of the CallMF package; they refer to the standard `modes.mf` distribution instead.

new or changed base files and the complete Computer Modern font family, the standard for most TEX users. You may add your own font programs here. METAFONT will find them when the environment variable `MFINPUTS` has an entry for this directory. (See section 4.3 for details about the environment variables recognized by METAFONT.)

`MetaFont/pool`: METAFONT (as well as TEX) is a PASCAL program, so it has to use a special method of string handling due to restrictions implied by this programming language. The 'pool' file `mf.pool` contains all texts and messages METAFONT is willing to print to your screen or the log file. It was automatically generated by `TANGLE` in the process of compilation and should not be deleted or changed in any way.

`MetaFont/rexx`: This directory contains the public domain package 'CallMF' (version 1.0), consisting of three ARexx scripts by Georg Heßmann, Martin Bokämper, Jörg Höhle, and Ulrich Wisser. It is very useful in connection with Heßmann's PasTEX. Both the previewer and the printer driver refer to 'CallMF' in case a font is missing at a certain size. The ARexx scripts will create the missing fonts automatically or in batch mode. You will find a documentation of 'CallMF' [1] in a subdirectory of `rexx`, so no more information about installation and usage of 'CallMF' is given at this place. Just a word about possible incompatibilities. In this distribution I use only modes from the standard distribution `modes.mf`, PasTEX relies on `pastex-modes.mf`. Unfortunately the naming convention in these two files is different, so some manual adjustment will be necessary.

4.2 Controlling memory sizes

Originally METAFONT (and TEX) is a PASCAL program not capable of dynamically controlling its memory allocation. Normally to change METAFONT's memory size you have to recompile the source code with changed internal settings. However, you need no longer fear the "`ask a wizard to enlarge me`" message, you can do some wizardry yourself! The present implementation is done with the help of the `Web2C` language converter and so it is possible to include a special feature for controlling the internal variables at runtime.

METAFONT uses an extra path and environment variable `MFCONFIG`. (See section 4.3 for further details on the environment.) The IniMF and VirMF programs will search these directories for a file called `mfmemory.config`, which is expected to contain instructions, one per line, of the form

```
set memmax       100000
set screenheight    400
set scalefactor       2
```

The example file given in the `config` directory 'sets' what are my present standard configuration, slightly higher than the internal defaults. Not all settings

need to be given in the file. Note that you can't just set everything to the maximum – that would require more than 8 Giga bytes of FastRAM![3]

If you create a standard base file with IniMF, any change to the configuration `mfmemory.config` will be acceptable for VirMF in connection with this specific base, except changes to `memtop`. This will result in the METAFONT message "`(Fatal base file error; I'm stymied)`," indicating that this base was produced with a different IniMF.

Here is a complete list of all parameters allowed in the configuration file, their internal defaults and some rules for their minimum and maximum values:

Setting	Default	Maximum	Purpose
`memmax`	30000	`maxhalfword`	Size of main storage
`maxinternal`	100	≤ 259774	Number of internals
`bufsize`	500	`maxhalfword`	Input characters
`errorline`	72	`maxhalfword`	Error context width
`halferrorline`	42	$<$`errline`-15	First error line
`maxprintline`	79	≥ 60	Output text line width
`screenwidth`	768	4095	Width of display screen
`screenheight`	1024	4095	Height of display screen
`stacksize`	30	300	Number of input sources
`maxstrings`	2000	`maxhalfword`	Number of strings
`poolsize`	32000	`maxhalfword`	Characters in strings
`movesize`	5000	10000	Storage for octant moves
`maxwiggle`	300	1000	Autorounded points per cycle
`gfbufsize`	100	4096	Size of the output buffer
`pathsize`	300	1000	Number of knots in a path
`bistacksize`	785	785	Stack for bisection algorithm
`headersize`	25	100	Number of TFM header words
`ligtablesize`	5000	32510	Ligature/kern steps
`maxkerns`	500	1000	Distinct kerns
`maxfontdimen`	50	100	Number of 'fontdimen's
`memtop`	30000	`memmax`	Largest memory index
`scalefactor`	1	10	Display is 'SF' times reduced
`maxinopen`	6	20	Number of input files
`paramsize`	150	1000	Macro parameters

Some of these values are internally multiplied by a factor of 4 or 8, so the actual number of bytes needed for a specific setting is a little bit hard to describe. As long as you don't encounter problems, I recommend that you stick to the defaults or the standard configuration file provided with this package. In most cases of insufficient memory size, the only value to increase is `memmax`, and this was actually tested up to the horribly high value 750,000 on my Amiga 2000 with 8 MB of 32 bit FastRAM.

[3] If you have that much, I don't believe you!

You may control the size of the online display by changing the values for `screenwidth`, `screenheight`, and `scalefactor`. The meaning of the first two should be clear, just keep in mind, that `screenheight` denotes the *interlaced* setting, although non-interlaced screenmodes will automatically be detected and taken care of, i.e., only half as much pixel lines will actually be drawn. And the two values don't set the size of the window, but the respective `innerwidth` and `innerheight`. The `scalefactor` is a reduction factor, i.e., the online display will be reduced by this factor in both its width and height. To make sensible use of this feature, the `plain.base` provided with this package was compiled with the internal settings

```
screen_rows:=4095; screen_cols:=4095; % infinity
```

defined in the printer `modes` file. When drawing on screen METAFONT uses the smaller value of `screenheight` and `screen_rows` respectively `screenwidth` and `screen_cols`, so actually the configuration overrides the internal settings.

4.3 Setting up the environment

METAFONT acknowledges a couple of environment variables in the course of its file searching algorithm, for the '`E`' feature to start your system editor in case of an error, and for setting up the online display. The IniMF and VirMF programs use multiple search paths when looking for command base files, the configuration file `mfmemory.config`, font description programs, and the text pool file `mf.pool`. Either you use the system call "`SetEnv <variable> <setting>`" to introduce your personal system configuration or you keep the internal defaults. Here is a list of the four path environments, their default settings and what METAFONT expects to find in the respective directories.

Environment	Default setting	Files
`MFBASES`	`",MF:bases"`	`*.base` dump files
`MFCONFIG`	`",MF:config"`	`mfmemory.config`
`MFINPUTS`	`",MF:inputs"`	`*.mf` program files
`MFPOOL`	`",MF:pool"`	`mf.pool` string file

The comma in these defaults denotes the current directory as the first entry, i.e., if you want to search in '`RAM:`' first, you could say `"RAM:,,MF:inputs"`, thus making the current directory the second search path. If the environment variables are not set by the 'SetEnv' function call the internal defaults will become active and the current directory is always searched first, so you may want to keep them.

In case of an error or an interruption, when METAFONT prompts you with a question mark you may answer by typing '`e`' (or '`E`'), thus invoking your text editor. This feature is controlled by the `MFEDIT` environment variable that holds a calling sequence to bring up your preferred system editor in the currently

processed file (not necessaryly the file you called METAFONT with) and in the offending line. Specify any program call or script that brings up your editor in file '`%s`' and line '`%d`.' The default setting is as follows, it invokes the `MEmacs` editor provided with the operating system:

```
EDITOR "MEmacs goto %d %s"
```

You will find some ARexx scripts suitable for the Cygnus Ed Professional editor in the `rexx` directory. If you want to use them, simply set the environment variable `MFEDIT` to the (string) value

```
"rx MF:rexx/MFEdit.rexx %s %d"
```

The second script `NameStruc` should be placed in the `REXX:` directory, because it is called directly from within `MFEdit.rexx`.

Also there is an online display feature as described in the 'METAFONTbook' by Don Knuth [4]. As it makes use of some advanced library functions not present in older versions of the Amiga operating system, it will work only with AmigaDOS 2.0 and higher (sorry). To activate this feature, set the environment variable `MFWTERM` to the value `"amiterm"` and every proof or smoke mode run will display its results on your screen.

5 Running the programs

5.1 Creating new bases with IniMF

Although there is a `plain.base` file in this package, you may want to create additional base files that include other METAFONT commands than the default set described in the 'METAFONTbook' [4]. For this purpose there is the IniMF program, capable of creating such binary bases for high-speed inclusion. If you call IniMF from the CLI (command line input, the shell) without an argument, it will prompt you with two asterisks, meaning "METAFONT is all geared up for action, please type a file name to continue."

```
> IniMF
This is METAFONT, C Version 2.71 (INIMF)
**
```

If you follow the concept proposed here, it is very easy to do the next step. Answer the METAFONT prompt with the name of an "initialization program" like `plain.inimf` or `cm.inimf` and the rest will be done automatically, resulting in a new base file `plain.base` or `cm.base`. Or you can type in consecutive `input` commands to load multiple files for inclusion in the base file. The last command should be `dump`; this will tell IniMF to create the compressed command file and to terminate.

Alternatively you may type a one-liner like

```
> IniMF plain.inimf
```

and the results will be exactly the same.

If you use `plain.inimf` or `cm.inimf` as proposed here, the standard distribution `modes.mf` (current version 1.2) by "Karl Berry and the METAFONT community" is added to your base file. This file contains mode descriptions for a variety of output devices, such as many matrix, laser or jet printers, photo typesetters, and screen previewers at different resolutions. There is a slight disadvantage when you include the complete `modes.mf` file, the base will heavily increase in its size. Feel free to extract only the modes you need for your personal configuration and to put them into a file `local.mf`. You should replace the line "`input modes`" by "`input local`" in `plain.inimf` and recreate the base file with the help of IniMF as described above.

PasTEX users who want to use the CallMF package may encounter some problems, because Georg Heßmann chose to create his own modes which are **not** compatible with `modes.mf`. These are collected in the file `pastex-modes.mf` in the `MF:rexx/callmf/inputs` directory. You can use these modes, but you will have to replace the entries in `modes` and recreate the base file to work with them. Or you can setup PasTEX to work with `modes.mf`.

5.2 Creating new fonts with VirMF

The production version of METAFONT is called 'VirMF' and is slightly different to IniMF. It is designed for high-speed production runs and is incapable of creating base files, but it can read such command bases. Just like with IniMF you have two possibilities to call VirMF. If you simply type its name (and if the operating system knows where to find the program), METAFONT will prompt you for an interactive dialog.

```
> VirMF
This is METAFONT, C Version 2.71
**
```

You may answer this prompt with any valid command, assuming 'plain' base preloaded as described in the 'METAFONTbook' [4].

For an easy beginning try the following (quoted from page 31 of the 'METAFONTbook' [4]). Type '`\relax`'—that's backslash, `r`, `e`, `l`, `a`, `x`—and hit ⟨return⟩ (or whatever stands for "end-of-line" on your keyboard). METAFONT is all geared up for action, ready to make a big font; but you're saying that it's all right to take things easy, since this is going to be a real simple run. The backslash means that METAFONT should not read a file, it should get instructions from the keyboard; the '`relax`' means "do nothing."

The machine will respond by typing a single asterisk: '`*`'. This means it's ready to accept instructions (not the name of a file). Type the following, just for fun:

```
drawdot (35,70); showit;
```

and ⟨return⟩—don't forget to type the semicolons along with the other stuff. A more-or-less circular dot should now appear on your screen! And you should also be prompted with another asterisk. Type

```
drawdot (65,70); showit;
```

and ⟨return⟩, to get another dot. (Henceforth we won't keep mentioning the necessity of ⟨return⟩ing after each line of keyboard input.) Finally, type

```
draw (20,40)..(50,25)..(80,40); showit;
shipit; end.
```

This draws a curve through three given points, displays the results, ships it to an output file, and stops. METAFONT should respond with '`[0]`', meaning that it has shipped out a character whose number is zero, in the "font" just made; and it should also tell you that it has created an output file called '`mfput.2602gf`'.

Alternatively you may specify one or more arguments on the command line when calling VirMF. The first argument may be the name of an alternative base file by prepending a `&` sign (as in `&plain`) and there may be a string of commands that will be executed after the base has been installed. For example, if you want to create a font at a specific resolution, type something like the following:

```
> VirMF &plain "\mode:=amiga; mag:=magstep2; input cmr10"
```

You have to type the quotation marks because of the semicolons. They have a special meaning for the Amiga operating system. This command will create the standard text font `cmr10` for the screen previewer at resolution 100 dpi but magnified twice, so the resulting font will get the name `cmr10.144gf` and METAFONT will write the following text to your screen (and into the log file).

```
This is METAFONT, C Version 2.71
(cmr10.mf (cmbase.mf) (roman.mf (romanu.mf [65] [66] [67] [68]
[69] [70] [71] [72] [73] [74] [75][76] [77] [78] [79] [80] [81]
[82] [83] [84] [85] [86] [87] [88] [89] [90]) (romanl.mf [97]
[98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108]
[109] [110] [111] [112] [113] [114] [115] [116] [117] [118]
[119] [120] [121] [122]) (greeku.mf [0] [1] [2] [3] [4] [5] [6]
[7] [8] [9] [10]) (romand.mf [48] [49] [50] [51] [52] [53] [54]
[55] [56] [57]) (romanp.mf [36] [38] [63] [62]) (romspl.mf [16]
[17] [25] [26] [27] [28]) (romspu.mf [29] [30] [31]) (punct.mf
[33] [60] [35] [37] [39] [40] [41] [42] [43] [44] [46] [47] [58]
[59] [61] [64] [91] [93] [96]) (accent.mf [18] [19] [20] [21]
[22] [23] [24] [32] [94] [95] [125] [126] [127]) (romlig.mf [11]
[12] [13] [14] [15]) (comlig.mf [34] [45] [92] [123] [124]) ) )
```

```
Font metrics written on cmr10.tfm.
Output written on cmr10.144gf (128 characters, 7276 bytes).
Transcript written on cmr10.log.
```

If you know a little bit of the METAFONT language, it should be clear what else you can do here; if you don't, I recommend that you buy the 'METAFONTbook' [4] or the second volume of Kopka's series [9].

5.3 Converting between gf- and pk-formats

METAFONT (and TEX) is a highly portable programming system available for a wide variety of computing machines from small systems like the Amiga to "number crunching monsters." Also its binary output is intended to be independent of a special hardware configuration. As in the example run listed in section 5.2, METAFONT normally produces threefold output files.

Typesetting systems like TEX don't know where to put the 'ink,' they only want information about the respective sizes of the characters and their positions in a font. These informations are written into the '`tfm`' file (`cmr10.tfm` in the example run) and contain the "(TEX) font metric" information.

Printer drivers or screen previewers that read "device independent files" produced by TEX or other programs need the actual information about the shapes of the characters they want to display. METAFONT writes the pixel information about every character in a font of type in the socalled "generic font format", i.e., it writes a '`gf`' file (`cmr10.144gf` in the example run). The file extension denotes the specific resolution of the produced font in "dots per inch," so you can distinguish between previewer fonts and printer fonts. The "generic font" format is very convenient at this place, and even some device driver programs can handle them directly.

More often your TEX system will deal with something called the '`pk`' representation of the pixel information. '`pk`' files contain the same information as their '`gf`' counterparts, but in a highly compressed format, so that '`pk`' files are much smaller. The conversion between the two pixel information formats is done by means of two programs, GFtoPK and PKtoGF. You call these programs in the following way

```
> GFtoPK [-v] gffile [pkfile]
> PKtoGF [-v] pkfile [gffile]
```

The '`-v`' option tells the programs to be a little more 'verbose', else you won't see any screen output. The first argument denotes the input file (don't leave the extension out) in the respective format. The optional second argument denotes the output file in the alternative format. If you don't give this second argument, the extension of the input file will be transformed in a standard way to the extension of the output file. The command line

```
> GFtoPK cmr10.144gf
```

will transform `cmr10.144gf` to `cmr10.144pk`. You may specify a full path name for the second argument and the output file will be created where you want it.

5.4 Fine control of the fonts with GFtoDVI

METAFONT knows two special modes 'proof' and 'smoke,' corresponding to virtual devices with a resolution of 2602 dpi. The program GFtoDVI can produce pretty printed hardcopies of fonts created for these devices. Simply type

```
> GFtoDVI
```

and GFtoDVI will prompt you for a `gf`-filename. You may change some internal settings for this run by appending a slash to this filename and GFtoDVI will prompt you for "special font substitutions," i.e., the replacement of 'gray' or 'black' fonts for a special output device. (See the 'METAFONTbook' [4] for details.) Like most other programs, GFtoDVI is capable of accepting its arguments directly from the command line input, so you may type

```
> GFtoDVI cmr10.2602gf/
```

and you will be prompted for further input.

5.5 Pretty printed font listings with MFT

Another special feature of this implementation is the MFT program that transforms ordinary METAFONT program source files into TEX input files for formatted output of the source listing. The only command line argument necessary is the name of a '`mf`'-file containing (parts of) a METAFONT program. You may influence the working of MFT by applying other formats than the `plain.mft`. It is fun to play with other settings and options.

```
> MFT file[.mf] [-cs] [change[.ch]] [style[.mft]]
```

5.6 Analyzing `gf`- and `pk`-files

The last two programs in this distribution are for font testing purposes. They will provide you with textual information about the contents of the pixel files in 'generic' and 'compressed' format.

```
> GFtype [-m] [-i] gffile
> PKtype pkfile
```

The two options for GFtype mean "Mnemonic output" and "pIxel output" respectively. This will add more information to the output, especially the outline of the characters in the form of ASCII 'pictures' for fine control of what METAFONT produces.

6 For further reading

Here is a short list of literature related to METAFONT and its companions. You should refer to these books when you seek for detailed information about programming in the METAFONT language. Also you may find very interesting stuff about the history of TEX and METAFONT since 1977, typesetting on computers in general, and the literate programming discipline used for the development of "Computers & Typesetting."

References

[1] Georg Heßmann et. al., ARexx Unterstützung für METAFONT, Dokumentation zur Version 1.0, Description of the 'CallMF' package by Georg Heßmann, Martin Bokämper, Jörg Höhle, and Ulrich Wisser for use with PasTEX.

[2] Donald E. Knuth, Computers & Typesetting A: The TEXbook, Addison-Wesley Publishing Company, 19th printing, June 1990. This is *the* reference manual for all users of the TEX system. It describes the TEX language in all levels of detail.

[3] Donald E. Knuth, Computers & Typesetting B: TEX: The Program, Addison-Wesley Publishing Company, fourth printing, February 1992. This 600 page book contains the lavishly documented and annotated source code for the TEX program, written in the WEB language.

[4] Donald E. Knuth, Computers & Typesetting C: The METAFONTbook, Addison-Wesley Publishing Company, fourth printing, June 1990. This is *the* reference manual for all users of the METAFONT system. It describes the METAFONT language in all levels of detail.

[5] Donald E. Knuth, Computers & Typesetting D: METAFONT: The Program, Addison-Wesley Publishing Company, fourth printing, February 1992. This 566 page book contains the lavishly documented and annotated source code for the METAFONT program, written in the WEB language.

[6] Donald E. Knuth, Computers & Typesetting E: Computer Modern Typefaces, Addison-Wesley Publishing Company, first printing, May 1986. This book contains specimen of all characters found in the Computer Modern font family, closely related to the appearance of all TEXnical documents.

[7] Donald E. Knuth, Literate Programming, Center for the Study of Language and Information, first printing, 1992. This is a collection of texts by Knuth related to the "literate programming discipline," applied to the highest extend in the TEX and METAFONT programs. Also it contains the errorlog for TEX 82 (until September 20, 1991).

[8] Helmut Kopka, LaTeX–Eine Einführung, Addison-Wesley Publishing Company, fourth printing, 1992. This is the first volume of the two-part series for German LaTeX users. It seems to be more popular than the reference manual by Leslie Lamport.

[9] Helmut Kopka, LaTeX–Erweiterungsmöglichkeiten mit einer Einführung in METAFONT, Addison-Wesley Publishing Company, third printing, 1992. This is the second volume of the two-part series for German LaTeX users. It contains an introduction to METAFONT and an introduction to literate programming in `WEB` and `CWEB`.

7 History

This Amiga port of METAFONT 2.71 is based in part on the complete TeX and METAFONT system supplied by Edmund Mergl on Fish Library Disks 611–616. Starting with METAFONT 2.7 from this implementation, I introduced a lot of modifications and enhancements. The major difference between the version by Edmund Mergl and my own is that now METAFONT is a complete stand-alone package, relying on self-contained source files and utilities. (See section 8.)

From the beginning I did all my programming on the Amiga with the SAS/C development system. The latest version 6.3 of this fine compiler is fully ANSI compliant, so it detected a list of omissions in style and speech when recompiling the source code. A major cleanup of the extra files for C, i.e., the introduction of prototypes and the correction of casting problems, was necessary.

As soon as version 2.7 was running, it was compared with the famous Amiga port by Stefan Becker from Fish Library Disk 486 and proved to be much slower, so some improvements seemed to be necessary to compete with this system as well. The tightest sections of the "inner loop" of METAFONT were recoded in assembly language for 68020 Motorola processors. This resulted in an overall speedup of about twenty to thirty percent compared to the non-optimized version and of about five to eight percent compared to the optimized version by Stefan Becker.

Running the TRAP test proved the correctness of the implementation, and this test routine by Donald Knuth was very convenient to find the last nasty bug in my own part (had something to do with a variable '`be_careful`'). But visual tests both with my own METAFONT and with Stefan Becker's port showed problems with the online display. Neither the implementation by Stefan Becker nor by Edmund Mergl worked in accordance with the 'METAFONTbook' [4]. Especially the test routine `6test.mf` provided for the Computer Modern font family caused some rethinking. (An example of this test can be found on page 192 of [4].) Finally a revised version of `amiga.c` was up and running, using some advanced functions for drawing on screen, thus requiring version 2 of the Amiga operating system.

Adapting some ideas by David Crooke implemented in his Amiga port of TEX 3.141, I introduced a revised path search algorithm and the possibility to use runtime-definable memory sizes with METAFONT. (See section 4.2 for details about the memory configuration.)

In September 1993 the first step to version 2.71 was done with the errata sheets provided by Addison-Wesley Publishing Company. All changes concerning the new version were implemented in the change file.

Finally in October 1993 I got hold of the official version 2.71[4], which replaced the old system completely. It included a new `plain.mf` base program and a whole new TRAP test. The last change made to this implementation was the introduction of a really BIG version of METAFONT (and of TEX 3.1415 by the way) by increasing `max_quarterword` to 32,767 (32 Kilo quarterwords) and `max_halfword` to 1,073,741,823 (1 Giga memwords), so now nobody will ever exceed the maximum setting for the main memory (you would have to have 8 Giga bytes of free memory!). Everybody seems to think that '262,142' is the maximum value for `memmax`, although this value holds for 36 bit words only. This C implementation works on the basis of "64 bit words," i.e., a halfword is defined as **long int**, so the very much higher value may be used.

8 Source code

This package is the "METAFONT user's kit," so no source code for the actual programs is provided with it (there is a directory of the complete Computer Modern font library, but this is a different kind of source code). A "developer's kit" is available from the implementor on request. If you would like to have the complete source code, send a self-addressed envelope and 10 Deutsche Mark (or 10 US Dollars) to the address on the cover of this documentation, and you will receive the latest version on disk together with all necessary tools for recompilation and future bug fixes.

[4] by anonymous ftp from the Stuttgart server `ftp.uni-stuttgart.de (129.69.8.13)`