COMPUTERS & TYPESETTING

JUBILEE • 2021 • edition

DONALD E. KNUTH

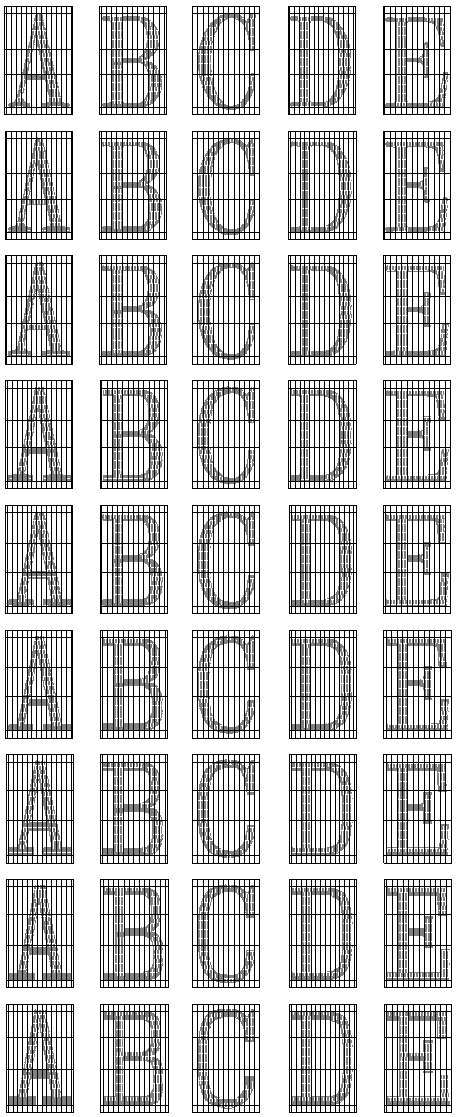
COMPUTERS & TYPESETTING / E

Computer Modern Typefaces

COMPUTERS & TYPESETTING / E

Computer Modern Typefaces

DONALD E. KNUTH *Stanford University*



ADDISON–WESLEY

Upper Saddle River, NJ
Boston · Indianapolis
San Francisco · New York
Toronto · Montréal
London · Munich
Paris · Madrid
Capetown · Sydney · Tokyo
Singapore · Mexico City

The quotations on pages 7 and 351 have been excerpted from the *Electra* file in the Dwiggins Collection of the Boston Public Library.

METAFONT is a trademark of Addison–Wesley Publishing Company.

TEX is a trademark of the American Mathematical Society.

The programs for Computer Modern are in the public domain, and readers may freely generate and hand-tune their own fonts using the algorithms of this book. However, use of the names is restricted: Any fonts whose names `cmr10` or `cmbx12` or ··· are identical to the standard font names of this book should be fully compatible with the fonts defined here; in other words, fonts with the same names are supposed to have precisely the same character coding schemes and precisely the same font metric files.

Library of Congress Cataloging-in-Publication Data

```
Knuth, Donald Ervin, 1938-
   Computer Modern typefaces.

   (Computers & Typesetting ; E)
   Includes indexes.
   1. Type and type-founding--Data processing.
2. Printing--Specimens.  3. METAFONT (Computer system).
4. Computerized typesetting.  I. Title.  II. Series:
Knuth, Donald Ervin, 1938-    .  Computers &
typesetting ; E.
Z253.3.K568  1986          686.2'2544      86-1235
ISBN 0-201-13446-2
```

Incorporates all corrections known in 2017.

Internet page `http://www-cs-faculty.stanford.edu/~knuth/abcde.html` contains current information about this book and related books.

Copyright © 1986 by Addison–Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publishers. For information regarding permissions, request forms, and the appropriate contacts with the Pearson Education Global Rights & Permissions Department, please visit `www.pearson.com/permissions/`. Printed in the United States of America.

ISBN-13 978-0-201-13446-9
ISBN-10 0-201-13446-2

Ninth printing, February 2021
Electronic edition, May 2021

To Martin E. Frost:
Who nurtured the systems

Preface

GREEK AND ROMAN LETTERFORMS, together with punctuation marks, numerals, and a large collection of mathematical symbols, are the subject of this book. Indeed, all of the characters printed in this volume were produced by the "programs" found here. By changing the parameters of these programs, it is possible to generate an extensive array of fonts, known as the Computer Modern family of typefaces.

Since this book is somewhat unusual, I had better try to explain what it contains and why a person might want to read it. Most of the following pages are filled with somewhat cryptic code in the METAFONT language, accompanied by annotated diagrams that illustrate what the METAFONT system might draw when presented with the given code. A preparatory chapter introduces the parameters of Computer Modern; this material explains how it is possible to generate many different alphabets from the same general specifications. The book concludes in a more traditional way, with samples of text showing the Computer Modern typefaces in use.

What is the value of such a book? Well, in the first place, it's fun just to look at mechanically constructed letters, even if the details of their generation aren't entirely understood. Secondly, people can easily make customized, one-time-only variations of Computer Modern for their personal use by learning how to assign new values to the parameters. Thirdly, people can make new symbols, or variations on the standard symbols, by mimicking the programs found here and making comparatively simple changes. Carefully crafted new symbols can be effective in many types of publications. Finally, this book provides hundreds of examples of METAFONT in use. The METAFONT system of typeface design is quite different from its predecessors, and a large collection of examples helps to demonstrate the principles that are involved.

The programs that appear in this book were developed over a period of years, and they have lost much of their original simplicity because of this long history of refinements. For example, the '*stem_corr*' parameter, which specifies slight adjustments to stem widths at high resolutions, was not part of the original design; hence many of the original programs were substantially shorter. Additional complications have been introduced by special instructions that round the lines and curves to desirable raster positions. Although the resulting METAFONT statements are not as straightforward as textbook examples would be, I think it is valuable to show the actual "optimized" code that has evolved; so I haven't held back any tricks of the trade that have contributed to the quality of

these fonts. These are the real programs, without the simplifications that were introduced for expository purposes in *The METAFONTbook*.

My colleague Charles Bigelow has kindly volunteered to write an introduction about "modern" fonts in general, so that the Computer Modern types can be understood in historical context. Although the so-called modern style of type is nearly 200 years old, I have always liked it because it was used in most of my favorite textbooks. Indeed, modern typefaces have been associated with technical publications of the highest quality for many generations. The Computer Modern designs are based to a considerable extent on the letterforms of "Monotype Modern 8A," which was used to typeset the first volumes of my books on *The Art of Computer Programming*.

The task of obtaining satisfactory typefaces with parametric variations has turned out to be much, much more difficult than I ever imagined when I began this work in 1977. But I've had the advantage of expert help; indeed, it has been my good fortune to make the acquaintance of several of the world's finest type designers, who have unselfishly given me a great deal of tutelage. At first I was blissfully unaware of the subtleties of the subject, because such things have rarely been described in print; when I completed the first "Computer Modern proto-type" in January 1980, I had been almost entirely self-taught, although I had spent much time in libraries looking for clues. Then in February of 1980, I had the great pleasure of demonstrating the first experimental METAFONT system to Hermann Zapf, who worked intensively with me for two weeks as we improved many of the character shapes. He taught me many things that I was able to use during the succeeding months. In 1981, I made several hundred specimen sheets of the fonts as they existed then, and I was privileged to receive detailed critiques of these specimens from Zapf in Germany as well as from Matthew Carter in England, and Charles Bigelow and Kris Holmes in America. Then Richard Southall came to Stanford during the month of April 1982, and we made extensive changes, especially to the sans-serif letters (for which he contributed a brand-new design). These changes spawned a completely new METAFONT language, which incorporated many ideas of Zapf, Bigelow, and Southall. Finally, the Computer Modern programs were rewritten again, in terms of the new METAFONT; the final draft, made during the spring of 1985, benefited greatly from detailed criticisms by Southall and Carter, as well as by N. N. Billawala, who also contributed a new design of the calligraphic letters. It is clear that I did not suffer from lack of expert assistance!

Another piece of luck came my way in 1984, when I learned that the original bronze patterns used to make the molds of Monotype 8A were in San Francisco. For years I had been working with indirect and imprecise information about the fonts that had stimulated this work. First I had worked from photographic blowups of letterpress original pages from *The Art of Computer Programming*; then Richard Southall had prepared enlargements from original proofs he had located in England. At last I found the actual 80-year-old patterns that had generated the metal type. The present owner of these patterns, Mr. Othmar Peters, kindly consented to let me borrow them while I was preparing the final draft of Computer Modern, and I learned much by measuring them with calipers. It never was my intention to make a slavish copy of any particular typeface, but the chance to work with these real artifacts made it possible to understand many of the intentions of the original designer.

If the Computer Modern typefaces have any merit, the credit should go to Zapf, Carter, Southall, Billawala, Bigelow, Holmes, and the anonymous original designer of Monotype 8A. I have tried to translate some of their wisdom into precise mathematical expressions. On the other hand, if these typefaces have deficiencies, the blame should be directed at me, because I did not always do what I was told.

I am dedicating this book to Martin E. Frost, who made it all possible by providing the WAITS and SAIL programming environments that have worked so efficiently for me during the past eight years. His yeoman service to Stanford has included countless rescue missions in the middle of the night, and he has introduced many system changes to accommodate the new equipment needed for our experiments.

The pleasant design of this book is due to Cleo Huggins, who found a nice way to fit illustrations and programs into the framework already established in *The METAFONTbook*.

I also want to reiterate my thanks to the many others acknowledged in previous volumes of this series—in particular, to my wife, Jill, and to my principal research sponsors, the National Science Foundation and the System Development Foundation.

Boston, Massachusetts — D. E. K.
February 1986

Modern Typefaces

Typography is a conservative art: The style that is called "modern" in English typographic terminology is two hundred years old. The first types in the modern style were cut late in the 18th century, toward the close of the age of enlightenment and the beginning of the industrial revolution. The principal originators of the style were Giambattista Bodoni, a printer and punch-cutter of Parma, and Firmin Didot, a typefounder member of an illustrious Parisian printing family. These were contemporaries of Thomas Jefferson, and if today we regard Jefferson as a man modern in many respects, it may not be too far-fetched to call the types of his time "modern" as well.

An alternative term for modern is "neoclassical." This places the style in the context of the classical revival in 18th century European art, and it avoids confusion with the 20th century avant-garde styles that are also called modern. However, typography is not simply a subset of the arts; the evolution of type forms is also part of the broader history of literacy.

To distinguish typography from the pictorial and plastic arts and to emphasize its unique achievements, many typographers now prefer a nomenclature devised by the French typographic savant, Maximilien Vox. In the Vox classification, the modern style is called "Didone," a neologism coined from the names of Didot and Bodoni. In paying homage to the original designers, the term reminds us that typefaces are public artifacts created with a personal vision.

Whatever we choose to call it, the modern style has several characteristic features. Analytically, these are:

- high contrast between thick and thin letter elements;
- rectangular stems and serifs without brackets;
- abrupt graduation from thin, elongate hairlines to thick bowls;
- emphasis on bilateral symmetry with a vertical axis in "shading" (or "stress"—the orientation of thick and thin elements);
- rectangularization of curved bowls toward cartouche-like shapes;
- reduction of variation in the widths of capital and lower-case letters;
- repeatability of component letter parts;
- regulated, sculptural treatment of details such as terminals, based on engraving rather than on writing.

The mæstro Bodoni himself tells us, more intuitively, that the main features of the modern face are: regularity; neatness and polish; good taste; and grace.

A modern type produces a brilliant texture on the page. Refined hairlines and serifs contrast with dark stems and bowls in a glittery chiaroscuro. Assimilated shapes and widths emphasize the vertical regularity of the letterforms. The emergent text pattern connotes an elegant sophistication.

After its beginnings in Italy and France, the modern style was established in England in 1800 by the London typefounder Robert Thorne, and soon after in Germany by the Weimar founder Justus Erich Walbaum. Modern types were popular throughout the 19th century, and were among the earliest adopted for mechanical composition equipment. The English Monotype Corporation produced Moderns No. 1, 7, 13, and 14 in its early days, and its American cousin firm produced Modern No. 8A, among others. The Monotype Modern families came to be liberally provided with additional characters, including the sorts necessary for mathematical and technical composition, hence the designs have had an enduring presence in scientific publishing.

In the 19th century, modern engendered a variant called Clarendon, similar in structure but with much heavier serifs and hairlines. Century Schoolbook, a text book face, and Corona, a newspaper text face, are 20th century versions of the Clarendon style in popular use today.

But typographic taste changes, and there have been significant reactions against modern. The Arts & Crafts printers of the turn of the century thought it degenerate, preferring to revive Medieval and Renaissance styles. Also rejecting modern as decadent, the proponents of the "new typography" championed sans-serif types in the 1920s and 30s.

Despite periodic reactions against it, the modern style has a stern beauty that continues to attract the attention of typographers even in our computer era. Revivals of the cuts of Bodoni, Didot, and Walbaum are available on most digital typesetting systems, and type designers have explored the idiom with fresh vision. Adrian Frutiger's Iridium is a lively evocation of the potentials of modern for photocomposition; Hermann Zapf's Marconi is an elegant modern designed for digital composition; and Donald Knuth's Computer Modern, the subject of this book, is a reinterpretation of Monotype Modern No. 8A.

In letterpress printing, modern fonts were technically troublesome because their delicate hairlines and serifs were particularly susceptible to damage; the broken letters produced a degraded text image. In digital typesetting with sufficient resolution, the delicate forms of modern can be rendered precisely. Computer typesetting and photolithographic printing are capable of reproducing

modern typefaces with a clarity and sharpness unobtainable at the time of the original development of the style.

Because writing preserves the forms of expression as well as the contents, we can maintain continuity with the literate images of past eras. In a type style like modern we can find inspiration to create letterforms that are contemporary and yet whose texture alludes to another time, another place, in much the way that a floral scent suffuses a winter room with the memory of a summer garden.

San Francisco, California — Charles A. Bigelow
January 1986

The modern or new-fashioned faced printing type at present in use
was introduced by the French, about twenty years ago:
the old shaped letters being capable of some improvement,
it was judged expedient to re-model the alphabet
to render them more agreeable to the improved state of printing;
but unfortunately for the typographic art,
a transition was made from one extreme to its opposite:
thus instead of having letters somewhat too clumsy,
we now have them with hair lines so extremely thin
as to render it impossible for them to preserve their delicacy
beyond a few applications of the lye-brush, or the most careful distribution;
thus may types be said to be in a worn state ere they are well got to work.
The hair lines being now below the surface of the main strokes of the letters,
the Printer, in order to get an impression of all parts of the face,
is obliged to use a softer backing, and additional pressure.

— RICHARD AUSTIN, *Specimen of Printing Types* (1819)

As far back as 1732, Fleischman,
the letter-engraver for the Enschedé Foundry of Holland,
gave more roundness to the round letters of the lower-case,
but his teachings found no imitative pupil.
Bodoni of Italy was more successful.
He increased the roundness suggested by Fleischman,
lengthened the ascending and descending strokes,
extended serifs, put more white space between the lines,
and thereby gave improved readability to his types.

— T. L. DE VINNE, *The Types of the De Vinne Press* (1907)

Bodoni's type displayed in a marked manner an attenuation of the thin lines,
with a reduction of the graduated portion of the curves to a minimum.
The letters are thereby weakened in construction
and turn a page into a maze of heavy lines
fretted here and there with grayness,
so that the eye is constantly readjusting its focus.
Morris says of it that it is the most illegible type ever cut,
with its preposterous thicks and thins;
he even speaks of "the sweltering hideousness of the Bodoni letter."

— F. W. GOUDY, *The Alphabet* (1918)

Likely enough it was Baskerville's 1773 edition of Ariosto's Orlando Furioso,
handsomely printed in Italian, which excited the admiration of Bodoni.
. . . Bodoni, however, cut several series of his own design;
he strengthened the thick lines and refined the thins,
thus producing that sharpness of contrast which,
exaggerated by the brilliance of his impression and the luxury of his paper,
seduced every European typographer from allegiance to
the more soundly built letters of obviously finer design
known to printers as the "old faces."

— STANLEY MORISON, *Four Centuries of Fine Printing* (1924)

The Didot modern face remained the standard letter in France,
and for the mass of books is still the normal design in use to-day.
Although some French printers have joined in the revival of old face,
the general typography has changed far less than in England.
A glance at any dozen recent novels printed in France
will show that the Didot tradition is by no means broken.

— A. F. JOHNSON, *Type Designs* (1934)

In pursuit of lively curves combined with a general "modern face" atmosphere,
we turned to one of the types that Bulmer used,
cut for him by William Martin around 1790—
and here seemed a good place to start again.
. . . The finishing strokes at the bottoms of letters,
cut straight across without "brackets,"
making sharp angles with the upright stems,
add "snap" to many of the old "modern face" designs—
and why not to Caledonia*?*

— W. A. DWIGGINS, *A New Printing Type, Caledonia* (1939)

Supplementary Bibliography

Readers who want to study more about typeface design with METAFONT may find a number of additional documents helpful. A complete bibliography of relevant literature would be extremely long; therefore this list has been limited to publications of the TEX project at Stanford University. Further information can be found in *TUGboat* (the journal of the TEX Users Group, published since 1980), and in the papers cited in the bibliographies of the papers cited here.

■ "Mathematical Typography" by Donald E. Knuth, *Bulletin of the American Mathematical Society* (new series) **1** (March 1979), 337–372. [Reprinted as part 1 of *TEX and METAFONT: New Directions in Typesetting* (Providence, R.I.: American Mathematical Society, and Bedford, Mass.: Digital Press, 1979).] *Discusses the author's motivation for starting to work on METAFONT, and illustrates the results achieved during the first six months of experiments.*

■ "The Computer Modern family of typefaces" by Donald E. Knuth, Stanford Computer Science Report 780 (Stanford, California, January 1980), 406 pp. *The first draft of what has developed into the present book and the present fonts. Includes complete programs written in the (now obsolete) METAFONT79 language, and discusses the early days of Computer Modern development.*

■ *The METAFONTbook* by Donald E. Knuth (Reading, Mass.: Addison–Wesley, 1986), xii+361 pp. [Also published as *Computers and Typesetting*, Volume C.] *The definitive user manual for METAFONT; a prerequisite for fully understanding the programs in the present book. Includes numerous examples, exercises, illustrations, lies, and jokes.*

■ "The concept of a meta-font" by Donald E. Knuth, *Visible Language* **16** (1982), 3–27. [See also pp. 308–359 for sixteen reviews of this paper.] *Illustrates the notion of variable parameters in typeface design.*

■ "Lessons learned from METAFONT" by Donald E. Knuth, *Visible Language* **19** (1985), 35–53. *Discusses the author's experiences with METAFONT79 and the early attempts at Computer Modern; illustrates several "meta-flops."*

■ "The letter S" by Donald E. Knuth, *The Mathematical Intelligencer* **2** (1980), 114–122. *Illustrates Renaissance approaches to this difficult letter, and develops the geometric theory that underlies the S's of Computer Modern.*

■ "An approach to type design and text composition in Indian scripts" by P. K. Ghosh, Stanford Computer Science report 965 (Stanford, California, April

1983), 148 pp. *Demonstrates that the methods developed for western languages and western alphabets work also for Devanagari and Tamil.*

■ "A Chinese meta-font" by Gu Guoan and J. D. Hobby, Stanford Computer Science report 974 (Stanford, California, July 1983), 22 pp.; reprinted in *TUGboat* **5**,2 (November 1984), 119–136. *Explains how to generate three different styles of Chinese characters satisfactorily by using only one program for each character; the different styles are obtained by defining nineteen general subroutines for basic strokes.*

■ "A course on METAFONT programming" by Donald E. Knuth, *TUGboat* **5**,2 (November 1984), 105–118. *Describes a series of 27 lectures by Charles A. Bigelow, Richard F. Southall, and the author about the history and frontiers of type design. These lectures were given while the implementation of METAFONT was being completed. Includes examples of numerous border designs as well as a "typeface" designed by the class participants.*

■ "Designing new typefaces with METAFONT" by Richard Southall, Stanford Computer Science report 1074 (Stanford, California, September 1985), 38 pp. *Presents a consistent terminology and a new typeface design created with* METAFONT*; evaluates the usefulness of symbolic programming languages as a tool for designers.*

■ *The Euler Project at Stanford* by David R. Siegel (Stanford, California: Department of Computer Science, 1985), ii+31 pp. *Explains how* METAFONT *was adapted to copy a large set of drawings by Hermann Zapf for the Euler typeface commissioned by the American Mathematical Society.*

■ "AMS Euler—A new typeface for mathematics" by Donald E. Knuth and Hermann Zapf, *Scholarly Publishing* **21** (1989), 131–157. *The story of a design project that helps bridge the gulf between mathematics and art.*

■ "Typesetting Concrete Mathematics" by Donald E. Knuth, *TUGboat* **10** (1989), 31–36, 342. *The parameters for Concrete Roman and Italic, a family of Computer Modern typefaces designed to complement AMS Euler.*

■ "Metamarks: Preliminary studies for a Pandora's Box of shapes" by Neenie Billawala, Stanford Computer Science report 1256 (Stanford, California, July 1989), 132 pp. *Lavishly illustrated studies in parameter variation, leading to the design of a new typeface family called Pandora.*

Contents

Introduction to the Parameters 1

Organization of the Files 8

Parameter Files 10

Driver Files 36

- `roman:` Roman text and typewriter text 36
- `title:` Caps and digits only 38
- `texset:` Extended ASCII character set 38
- `csc:` Caps and small caps 40
- `textit:` Italic text and typewriter text 41
- `mathit:` Math italic 42
- `mathsy:` Math symbols 44
- `mathex:` Math extension characters 44

Program Files 46

- `accent:` Accents common to roman and italic 49
- `bigacc:` Wide accents for math extension font 63
- `bigdel:` Delimiters for math extension font 66
- `bigop:` Operators for math extension font 103
- `calu:` Calligraphic capitals 123
- `comlig:` Ligatures common to roman and italic 141
- `cscspu:` Special uppercase for caps and small caps . . . 145
- `greekl:` Lowercase Greek 147
- `greeku:` Uppercase Greek 171
- `itald:` Italic digits 193

`italig:` Italic f-ligatures 199
`itall:` Italic lowercase 209
`italms:` Italic math specials 233
`italp:` Italic punctuation 239
`italsp:` Special lowercase for italic 245
`olddig:` Oldstyle digits 251
`punct:` Punctuation common to roman and italic . . 265
`romand:` Roman digits 285
`romanl:` Roman lowercase 305
`romanp:` Roman punctuation 361
`romanu:` Roman uppercase 369
`romlig:` Roman f-ligatures 421
`romms:` Roman math specials 431
`romspl:` Special lowercase for roman 443
`romspu:` Special uppercase for roman 451
`romsub:` Substitutes for ligatures 457
`sym:` Math symbols common to several fonts . . 459
`symbol:` Math symbols in symbol font only 483
`tset:` Extended ASCII symbols 542
`tsetsl:` Extended ASCII symbols to be slanted . . . 543

The Base File 545
Font Specimens 554
General Index 571
Index to Character Programs 581

Grandjean, Baskerville, Bodoni and the Didots
had a mischievous influence on type-forms;
for the derivations *from types that their work made popular culminated in*
a kind of letter which was capable of greater vulgarity and degradation
than was ever the case with older fonts.

— D. B. UPDIKE, *Printing Types* (1927)

Introduction to the Parameters

Zillions of alphabets can be generated by the programs in this book. All you have to do is assign values to 62 parameters and fire up the METAFONT system; then presto—out comes a new font of type.

The purpose of this introductory chapter is to explain the significance of each of the 62 parameters that define a Computer Modern typeface. The parameters divide naturally into several groups.

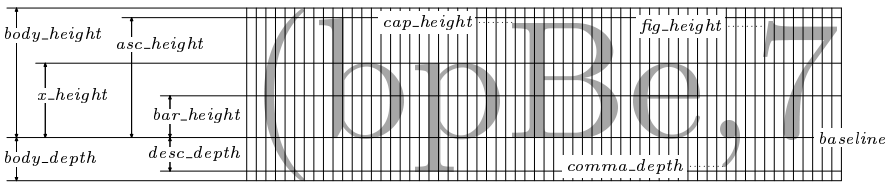
First come vertical measurements that define the heights and depths of most of the characters:

- *body_height* is the height of parentheses and other symbols like '/';
- *asc_height* is the height of lowercase letters with ascenders (b d f h k l);
- *cap_height* is the height of uppercase letters (A B C $\cdots$ Z);
- *fig_height* is the height of numerals (0 1 2 $\cdots$ 9);
- *x_height* is the height of lowercase letters without ascenders (a c e g m $\cdots$);
- *desc_depth* is the depth of lowercase letters with descenders (g j p q y);
- *comma_depth* is the depth of comma, semicolon, and 'Q';
- *math_axis* determines the middle of parentheses and certain other symbols (/ = + $\cdots$);
- *bar_height* determines many things, including the position of the bar line in 'e' and 'æ'.

Some characters have heights and depths determined indirectly from these quantities and other parameters; for example, the height of 'i' and 'j' is

$$\min\bigl(asc_height,\ \tfrac{10}{7}x_height + .5\max(dot_size,\ cap_curve)\bigr)$$

and the height of 't' is $\min(asc_height,\ \frac{10}{7}x_height)$. The diagrams of characters in this book have horizontal lines that show *body_height*, *asc_height*, *x_height*, *bar_height*, the baseline, *desc_depth*, and *body_depth*, where *body_depth* exceeds *desc_depth* by the same amount that *body_height* exceeds *asc_height*.



The *bar_height* parameter is somewhat delicate, because the author used it too frequently in his first designs of 1977; he never had the courage to abandon this early idea. If you increase *bar_height*, certain features of the lowercase letters will move upward in possibly unexpected ways; in fact, *bar_height* even influences characters like '4', 'G', and '&'! Therefore it is safest to keep *bar_height* between about 50% and 55% of *x_height*. The other height-and-depth parameters can be varied more freely.

Character shapes often "overshoot" the stated heights and depths. For example, an 'e' actually descends slightly below the baseline and ascends slightly above the *x_height*; this makes it visually compatible with letters like 'x' that do not overshoot. Two parameters are involved:

- *o* governs the overshoot of curved lines like the top and bottom of 'O';
- *apex_o* governs the overshoot at corners like the top of 'A' and the bottom of 'V'.

(The actual amount of overshoot varies from character to character. For example, letters like 'C', 'O', and 'S' overshoot their nominal height and depth by o, but 'e' and 'o' rise only $.75o$ above *x_height* and descend only $.5o$ below the baseline. The letters 'V' and 'W' dip below the baseline by *apex_o*, but 'v' and 'w' go down only half as much.) Small corrections like this are generally adjusted last, after the more significant measurements of a font have been established.

The widths of characters are determined primarily by a ubiquitous parameter that retains traces of its Monotype heritage:

- u is the basic unit of width.

There are essentially $18u$ to an em in the Computer Modern designs, and most characters have a prescribed width that is a multiple of u. For example, 'A' is $13u$ wide, and 'a' is $9u$; the widths of the fundamental stem-letters 'l', 'n', and 'm' are respectively $5u$, $10u$, and $15u$.

Characters aren't constrained to be an integer number of units wide, as they were in Monotype casting machines; but the old system provides a good reference point, and u is still a convenient unit to work with. The vertical lines in all character diagrams of this book are one unit apart. Increasing the value of u makes a font more extended; decreasing u makes it more condensed.

The proper fitting of letters is a subtle problem, hence character widths don't depend on u alone. Four additional parameters are available for finer adjustments:

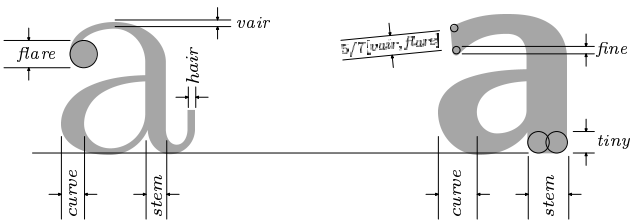
- *serif_fit* is added to intercharacter spacing where lowercase serifs appear;
- *cap_serif_fit* is added to intercharacter spacing where uppercase serifs appear;
- *letter_fit* is added to the intercharacter spacing at the left and right of every character;
- *width_adj* is added to the widths of some letters (H M N) and subtracted from the widths of others (E F L O Q T).

(The *width_adj* attempts to reconcile competing notions about how wide certain letters ought to be; it is usually zero in a font with serifs, and between 0 and u in a font without serifs.) A sans-serif font usually has negative values for *serif_fit* and *cap_serif_fit*, to compensate for the fact that serifs are not present; on the other hand, the font `cmr10` used in this sentence has *serif_fit* = 0, and its *cap_serif_fit* is slightly positive. A positive *letter_fit* puts more space between letters; this is often appropriate for small sizes of type. A negative *letter_fit* makes letters touch or nearly touch; this is sometimes appropriate in large title fonts.

The darkness or "weight" of a Computer Modern character is affected by quite a few parameters, of which the most important for lowercase letters are called *hair*, *vair*, *stem*, *curve*, and *flare*:

- *hair* is the horizontal breadth of thin "hairline" strokes;
- *vair* is the vertical breadth of thin "hairline" strokes;
- *stem* is the breadth of straight stem lines;
- *curve* is the breadth of curved stem lines;
- *flare* governs the diameter of bulbs or other terminal strokes.

A "monoline" font in which there is little or no contrast between thick and thin strokes often has *hair* = *stem* = *curve*, although *vair* is usually a bit smaller than this common value. The `cmr10` font you are now reading has *hair* = $^{9}/_{36}$ pt, *vair* = $^{8}/_{36}$ pt, *stem* = $^{25}/_{36}$ pt = $1.25u$, *curve* = $^{30}/_{36}$ pt = $1.5u$, and *flare* = $^{33}/_{36}$ pt; here *curve* exceeds *stem*, to give the illusion of equal weight.



Computer Modern fonts with serifs have "bulbs" on letters like 'a', 'c', 'f', etc.; these bulbs are replaced in sans-serif fonts by slightly flared terminal strokes. The weight is controlled in both cases by the stated amount of *flare*, often in combination with *vair*.

Several other weight-oriented parameters are also necessary:

- *thin_join* controls thickness in critical parts of 'e', 'h', etc.;
- *bar* is the thickness of the crossbar in 'f' and 't';
- *slab* is the vertical thickness of serifs and arms;
- *dot_size* is the diameter of dots in 'i', ':', '!', etc.;
- *ess* is the stroke breadth in the middle of 's';
- *rule_thickness* is the breadth of strokes in mathematical symbols like '+', '/', and '='.

The *slab* value affects all serifs, and it is important also in sans-serif fonts because it governs the thickness of so-called arms in letters like 'E'. An independent *ess* parameter seems to be necessary even though it affects only one letter, because subtle combinations of *x_height*, *u*, *hair*, *vair*, and *curve* seem to influence the apparent thickness of this special stroke; the *ess* value should be fine-tuned after the rest of the font is in place.

Uppercase letters use the lowercase values of *vair*, *flare*, and *slab*, but they have their own parameters for other kinds of weights:

- *cap_hair* is the horizontal breadth of uppercase hairlines;
- *cap_stem* is the breadth of straight uppercase stem lines;
- *cap_curve* is the breadth of curved uppercase stem lines;
- *cap_ess* is the stroke breadth in the middle of 'S';
- *cap_bar* is the thickness of bar lines in 'E', 'F', 'G', 'H';
- *cap_band* is the thickness of horizontal strokes in 'A', 'B', 'D', 'P', 'R'.

There's not really a clear-cut distinction between *cap_bar* and *cap_band*, which are equal to each other in most of the standard Computer Modern fonts; the sans-serif case seems to work best if $slab \geq cap_bar \geq cap_band$.

If all stems are made exactly the same breadth, some of them will appear to be darker than others, because our eyes are influenced by other parts of the character shapes. Therefore small corrections are made:

- *stem_corr* controls small refinements to stems;
- *vair_corr* controls small refinements to hairlines.

For example, *stem_corr* is taken away from the stem inside the lobe of the 'P' (but the lower part of that stem is left untouched). The thickness at the top of 'O' is increased by *vair_corr*, and it is increased by $1.5\,vair_corr$ at the bottom.

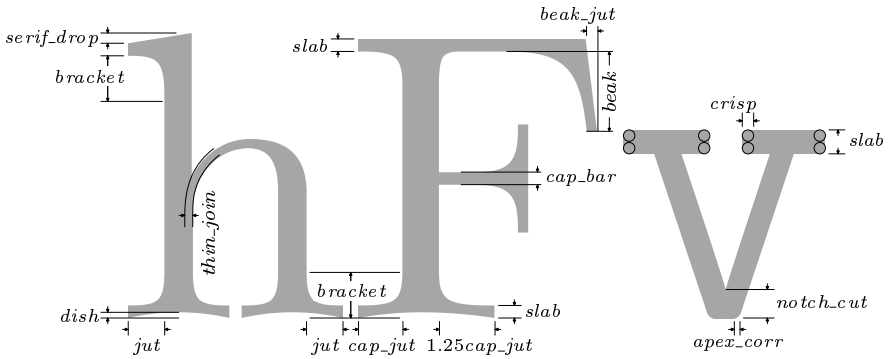
Three important parameters are available to control the "softness" of corners in the letter shapes:

- *crisp* is the diameter of serif corners;
- *tiny* is the diameter of rounded corners;
- *fine* is the diameter of sharply rounded corners.

For example, setting $crisp = 0$ gives perfectly square corners to serifs (as in `cmr10`); but if $crisp = slab$, the serifs end in semicircles (as in `cmtt10`).

Serifs and "arms" can be varied in several ways by diddling with another set of parameters:

- *jut* is the amount by which lowercase serifs protrude;
- *cap_jut* is the amount by which uppercase serifs protrude;
- *serif_drop* is the amount by which certain serifs slope;
- *dish* is the amount erased at top or bottom of serifs;
- *bracket* is the vertical distance from serif to stem tangent;
- *beak* is the vertical protrusion of beak serifs at the ends of arms;
- *beak_jut* is the horizontal protrusion of beak serifs.



When diagonal strokes come together, the Computer Modern routines have two ways to keep the junction from filling in and looking too dark: The strokes are first spread apart by *apex_corr*; then the inner edge is opened, if necessary, until the height of the intersecting strokes is *notch_cut* or less.

- *apex_corr* is the extra external stroke width at diagonal junctions;
- *notch_cut* is the maximum vertical thickness at lowercase diagonal junctions;
- *cap_notch_cut* is the uppercase counterpart of *notch_cut*.

In `cmr10` the *apex_corr* is zero, and *notch_cut* is 10 pt; this effectively disables such corrections, since they are unnecessary in a font with strong contrast between thick and thin strokes. However, `cmtt10` has *apex_corr* $= {}^{10}/_{36}$ pt and *notch_cut* $= {}^{25}/_{36}$ pt.

The 48 parameters named so far have all been "ad hoc dimensions," which must be specified as physical units of measure in terms of inches, centimeters, or points, etc. The next several parameters are different; they are "pure numbers," i.e., dimensionless quantities.

- *slant* is the change in horizontal position per unit of vertical distance from the baseline, in italic or oblique fonts.

For example, the fonts `cmr10` and `cmsl10` are generated by exactly the same parameter values, except that `cmr10` has *slant* $= 0$ *while* `cmsl10` *has slant* $= 1/6$.

- *fudge* is a factor applied to the stem thicknesses of certain "heavy" characters.

For example, the stroke widths of 'w' are based on *fudge* $\times$ *hair* and *fudge* $\times$ *stem*, not simply on *hair* and *stem*. Font `cmr10` has *fudge* $= 1$, so the fudge factor has no effect; but `cmtt10` has *fudge* $= 0.81$.

- *math_spread* tells how "open" certain mathematical symbols are.

For example, `cmr10` has *math_spread* $= 0$, `cmr9` has *math_spread* $= 1/5$, ..., `cmr5` has *math_spread* $= 1$; this makes more space between the bars of '=' signs and spreads out several other symbols in small sizes of type.

- *superness* is the parameter used in superelliptical bowls.

For example, the symbols '0', '0', and '0' have respective supernesses of 0.71, 0.77, and 0.84. (Chapter 14 of *The METAFONTbook* explains more about *superness*.)

- *superpull* is an amount by which superelliptical bowls are thinned.

The inside of the bowl is pulled toward the outside by this factor.

- *beak_darkness* governs the thickness of beaks.

The inside edge of a beak moves outward as *beak_darkness* increases. Font `cmr10` has *beak_darkness* = 11/30, while `cmtt10` has *beak_darkness* = 0.

- *ligs* controls the number of ligatures in the font.

If *ligs* = 0 you get only Spanish punctuation marks as ligatures; if *ligs* = 1 you get also double quotes and dashes; if *ligs* = 2 you get also the f-ligatures 'ff', 'fi', 'fl', 'ffi', and 'ffl'.

We have now covered 55 of the 62 parameters. The remaining seven all have true/false values; i.e., they're boolean, not numeric.

- *serifs* tells whether serifs and bulbs should be attached;
- *square_dots* tells whether dots should be square, not rounded;
- *hefty* tells whether weight-reducing strategies should be used;
- *monospace* tells whether the characters should all be forced to have the same width;
- *variant_g* tells whether 'g' should be used instead of 'g';
- *low_asterisk* tells whether '∗' should be used instead of '*';
- *math_fitting* tells whether to produce special spacing for TEX math mode.

These parameters are fairly straightforward, except for '*hefty*'. When *hefty* is *true*, several characters are drawn in a special style that tends to avoid clusters of heavy strokes.

It's not easy to describe all the effects of each parameter. The index to this book should be consulted if you want to discover every place in which a certain parameter is used in the designs.

Many of the 62 parameters depend on each other in subtle ways, so you cannot expect to obtain a very good font if you simply select 62 values at random. The proper balance between *stem* and *curve* is particularly important. Several dozen combinations of values are given in this book as examples of possibilities that do seem to work; and thousands of variations that are reasonably near these "standard" values can be expected to produce satisfactory results. But if you try parameter settings that differ widely from those the author actually had time to examine while writing these programs, you will probably encounter cases in which individual shapes need to be adjusted.

The parameter values must, in fact, satisfy certain minimum conditions, or disaster will almost certainly ensue. The following relations are implicitly assumed to hold in all applications of the Computer Modern routines:

1) The *asc_height* must be larger than the *x_height*. Indeed, *asc_height* should be enough bigger that there is room for accents and for the distinctive features of letters like 'f', 'i', 'j', 't'.

2) The stem weights should satisfy

$$curve \geq stem; \qquad cap_stem \geq stem; \qquad cap_curve \geq curve.$$

3) The stem weights must not be less than the corner diameters. More precisely, each of the values

$$hair,\ vair,\ stem,\ curve,\ ess,\ flare,\ dot_size,\ bar,\ slab,$$
$$cap_hair,\ cap_stem,\ cap_curve,\ cap_ess,\ cap_bar,\ cap_band$$

must be greater than or equal to each of the values

$$crisp,\ tiny,\ fine;$$

and *thin_join* should not be less than *fine*.

4) The "second-order refinement" values *stem_corr* and *vair_corr* should be very small. More precisely, *stem_corr* should not exceed

$$\tfrac{1}{5}cap_hair, \qquad \tfrac{1}{6}stem, \qquad \tfrac{1}{4}fudge \times stem, \qquad \tfrac{1}{12}curve;$$

and *vair_corr* should not exceed $\frac{1}{4}slab$. If *hefty* is *true*, the stricter relation

$$stem_corr \leq \tfrac{1}{16}fudge \times hair$$

should in fact be valid.

Trouble might also arise if the device-specific parameter called *blacker* is made negative. (The *blacker* value is not a parameter of Computer Modern, but it is a parameter of plain METAFONT, which is assumed to underlie the Computer Modern routines; see Chapter 11 of *The METAFONTbook*.) Since *blacker* is added to all stem weights behind the scenes, and since it is not added to the corner diameters, a relation like '*thin_join* $\geq$ *tiny*' might be violated when *blacker* < 0.

The programs will fail at extremely low resolutions, when there are fewer than 100 or so pixels per inch; some of the key points will be too close together, if there aren't enough pixels to go around. However, the standard fonts have all been tested successfully at a resolution of 101 per inch.

Electra Italic, really an oblique roman—
and the first to be available for machine composition in America—
is so readable a letter, that entire texts may readily be set in it.
— MERGENTHALER LINOTYPE COMPANY, *Electra* (1936)

A publisher called me up today and suggested that
a bold version of the original Electra Italic (oblique roman)
would make a wonderful letter in its own right,
and inquired if we had any intention of cutting it.
— C. H. GRIFFITH, *Letter to W. A. Dwiggins* (1941)

Organization of the Files

The programs for Computer Modern consist of hundreds of little pieces that are grouped together in dozens of files. There are four main types of files:

- The *parameter files* define parameters for specific typefaces.
- The *driver files* control the font-generation process and define ligature and kerning conventions.
- The *program files* contain short programs for individual characters.
- The *base file* contains macros and declarations that are used by all the programs.

There also are *utility files* that facilitate the testing of new characters; these are described in Appendix E of *The METAFONTbook*.

When you want to generate a new font for a particular device (e.g., for the hypothetical *luxo* printer described in *The METAFONTbook*), you simply invoke METAFONT with a command line like

```
\mode=luxo; input cmr10
```

and stand back. The machine will begin by reading the parameter file `cmr10.mf`; then `cmr10.mf` will call in the base file `cmbase.mf`, followed by a driver file called `roman.mf`. The driver file, in turn, will input program files like `romanu.mf`, generating characters one by one; your terminal will look something like this:

```
(cmr10.mf (cmbase.mf) (roman.mf (romanu.mf
The letter A [65]
The letter B [66]
  ⋮
```

A verbose printout has appeared because *luxo* mode sets *tracingtitles* := 1, according to Chapter 11 of *The METAFONTbook*. If you're using a mode for which *tracingtitles* = 0, you won't see titles like '`The letter A`'; in this case your screen will say

```
(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]) ) )
```

at the end of the run. The driver file `roman.mf` has input 11 program files called `romanu.mf`, `romanl.mf`, `greeku.mf`, `romand.mf`, `romanp.mf`, `romspl.mf`, `romspu.mf`, `punct.mf`, `accent.mf`, `romlig.mf`, and `comlig.mf`.

Other driver files—such as `textit.mf`, which is used for text italic fonts like `cmti10`—will input different combinations of program files. For example, both `roman.mf` and `textit.mf` make use of `romanu.mf` for uppercase letters; but roman style lowercase letters come from `romanl.mf`, while the italic style comes from another program file called `itall.mf`.

In this book we shall examine the contents of all the files. First we'll look at parameter files (which are the easiest); then we'll consider the eight standard driver files (which are necessary but boring); and then we will get to the real meat of the matter, the 30 program files that describe approximately 500 characters. Parts of the base file `cmbase.mf` will be introduced together with the program files, since subroutines are most easily understood when they can be seen in use. Finally we'll wrap everything up by listing and discussing the remaining parts of `cmbase.mf`.

A comprehensive index at the back of this book makes it easy to find the uses of all the variables and the definitions of all the macros. There's also an index that takes you from character shape and/or font position to the relevant program page.

If you want to generate a new Computer Modern font with personalized parameters, you simply need to create a new parameter file analogous to the files found in this book. You might also want to go further and make your own driver file and/or program files, analogous to those presented here. Nothing in this book is "sacred," although you're not supposed to use the standard font names (`cmr10`, `cmtt10`, etc.) after you have made changes.

Parameter Files

There are altogether 75 standard parameter files in the Computer Modern family, hence there are 75 standard fonts. The "most standard" of all is `cmr10`, since the 10-point roman characters of Monotype Modern 8A served as the original starting point for all the Computer Modern designs.

Here is a transcript of the file `cmr10.mf`, which is typical of all the others. Notice that it defines the 62 parameters we discussed earlier, and it also does a little more: It prescribes a **font_identifier**, `"CMR"` (which characterizes a subfamily of Computer Modern that is useful in some schemes for font library maintenance); and it specifies a **font_size** of 10 points (so that, for example, TEX users can ask for '`cmr10 at 20pt`' if they want two-fold magnification).

```
% Computer Modern Roman 10 point
if unknown cmbase: input cmbase fi                    % ensure that cmbase is present

font_identifier := "CMR"; font_size 10pt#;

u# := 20/36pt#;                                       % unit width
width_adj# := 0pt#;                                   % width adjustment for certain characters
serif_fit# := 0pt#;                                   % extra sidebar near lowercase serifs
cap_serif_fit# := 5/36pt#;                            % extra sidebar near uppercase serifs
letter_fit# := 0pt#;                                  % extra space added to all sidebars

body_height# := 270/36pt#;                            % height of tallest characters
asc_height# := 250/36pt#;                             % height of lowercase ascenders
cap_height# := 246/36pt#;                             % height of caps
fig_height# := 232/36pt#;                             % height of numerals
x_height# := 155/36pt#;                               % height of lowercase without ascenders
math_axis# := 90/36pt#;                               % axis of symmetry for math symbols
bar_height# := 87/36pt#;                              % height of crossbar in lowercase 'e'
comma_depth# := 70/36pt#;                             % depth of comma below baseline
desc_depth# := 70/36pt#;                              % depth of lowercase descenders

crisp# := 0pt#;                                       % diameter of serif corners
tiny# := 8/36pt#;                                     % diameter of rounded corners
fine# := 7/36pt#;                                     % diameter of sharply rounded corners
thin_join# := 7/36pt#;                                % width of extrafine details
hair# := 9/36pt#;                                     % lowercase hairline breadth
stem# := 25/36pt#;                                    % lowercase stem breadth
curve# := 30/36pt#;                                   % lowercase curve breadth
ess# := 27/36pt#;                                     % breadth in middle of lowercase 's'
flare# := 33/36pt#;                                   % diameter of bulbs or breadth of terminals
```

```
dot_size# := 38/36pt#;                    % diameter of dots
cap_hair# := 11/36pt#;                    % uppercase hairline breadth
cap_stem# := 32/36pt#;                    % uppercase stem breadth
cap_curve# := 37/36pt#;                   % uppercase curve breadth
cap_ess# := 35/36pt#;                     % breadth in middle of uppercase 's'
rule_thickness# := .4pt#;                 % thickness of lines in math symbols

dish# := 1/36pt#;                         % amount erased at top or bottom of serifs
bracket# := 20/36pt#;                     % vertical distance from serif base to tangent
jut# := 28/36pt#;                         % protrusion of lowercase serifs
cap_jut# := 37/36pt#;                     % protrusion of uppercase serifs
beak_jut# := 10/36pt#;                    % horizontal protrusion of beak serifs
beak# := 70/36pt#;                        % vertical protrusion of beak serifs
vair# := 8/36pt#;                         % vertical diameter of hairlines
notch_cut# := 10pt#;                      % maximum breadth above or below notches
bar# := 11/36pt#;                         % lowercase bar thickness
slab# := 11/36pt#;                        % serif and arm thickness
cap_bar# := 11/36pt#;                     % uppercase bar thickness
cap_band# := 11/36pt#;                    % uppercase thickness above/below lobes
cap_notch_cut# := 10pt#;                  % max breadth above/below uppercase notches
serif_drop# := 4/36pt#;                   % vertical drop of sloped serifs
stem_corr# := 1/36pt#;                    % for small refinements of stem breadth
vair_corr# := 1/36pt#;                    % for small refinements of hairline height
apex_corr# := 0pt#;                       % extra width at diagonal junctions

o# := 8/36pt#;                            % amount of overshoot for curves
apex_o# := 8/36pt#;                       % amount of overshoot for diagonal junctions

slant := 0;                               % tilt ratio (Δx/Δy)
fudge := 1;                               % factor applied to weights of heavy characters
math_spread := 0;                         % extra openness of math symbols
superness := 1/sqrt 2;                    % parameter for superellipses
superpull := 1/6;                         % extra openness inside bowls
beak_darkness := 11/30;                   % fraction of triangle inside beak serifs
ligs := 2;                                % level of ligatures to be included

square_dots := false;                     % should dots be square?
hefty := false;                           % should we try hard not to be overweight?
serifs := true;                           % should serifs and bulbs be attached?
monospace := false;                       % should all characters have the same width?
variant_g := false;                       % should an italic-style 'g' be used?
low_asterisk := false;                    % should the asterisk be centered at the axis?
math_fitting := false;                    % should math-mode spacing be used?

generate roman                            % switch to the driver file
```

All of the dimension-oriented parameters have been given in terms of "sharped" units (involving *pt#*), according to the conventions of plain METAFONT. The **font_setup** routine in `cmbase` will convert device-independent values such as *u#* and *hair#* into pixel-oriented values like *u* and *hair*.

The final line of `cmr10.mf` tells METAFONT to '**generate** `roman`'. This normally calls in the driver file `roman.mf`, which takes charge of the remaining details of font generation.

Most of the parameter values of `cmr10` are given as multiples of $1/36$ pt. There's no good reason for this, except that the author's original sketches in 1977 were made on graph paper with 36 pixels per point; the same resolution was then used to make proofs throughout most of the development process. When it became desirable to make stems one pixel darker or one pixel lighter on experimental proofsheets, the corresponding parameter was simply increased or decreased by 1/36 of a point.

All of the Computer Modern parameter files look like `cmr10.mf`, except for changes in name and parameter values. Therefore it will suffice to specify the other 73 files in an abbreviated, tabular format.

Computer Modern Roman types appear in eight standard sizes. The largest of these, 17.28 pt, corresponds to a magnification of 10-point type by a factor of '`\magstep3`' in the conventions of plain TEX.

name	`cmr17`	`cmr12`	`cmr10`	`cmr9`	`cmr8`	`cmr7`	`cmr6`	`cmr5`
font_identifier	`CMR`	`CMR`	`CMR`	`CMR`	`CMR`	`CMR`	`CMR`	`CMR`
font_size	17.28	12	10	9	8	7	6	5
u	32.5	23.5	20	18.5	17	15.5	14	12.5
width_adj	0	0	0	0	0	0	0	0
serif_fit	0	0	0	0	0	0	0	0
cap_serif_fit	8	5.8	5	4.6	4.1	3.5	2.8	2
letter_fit	−0.1	0	0	0	0	2	3	5
body_height	466	324	270	243	216	189	162	135
asc_height	432	300	250	225	200	175	150	125
cap_height	425	295.2	246	221.4	196.8	172.2	147.6	123
fig_height	401	278.4	232	208.8	185.6	162.4	139.2	116
x_height	267.8	186	155	139.5	124	108.5	93	77.5
math_axis	155.5	108	90	81	72	63	54	45
bar_height	150	104.4	87	78.3	69.6	60.9	52.2	43.5
comma_depth	121	84	70	63	56	49	42	35
desc_depth	121	84	70	63	56	49	42	35
crisp	0	0	0	0	0	0	0	0
tiny	8	8	8	8	8	7	7	6
fine	7	7	7	7	7	6	6	5
thin_join	9	7.5	7	7	7	6	6	5
hair	11	9.5	9	9	9	8.5	8	7.5
stem	33	28	25	24	22	20	18	16
curve	40	33	30	28	25.5	23	20.5	18
ess	35	30	27	25	23	21	17	12
flare	45	36	33	30	27	24	21	19

dot_size	51	42	38	35	32	29	26	22
cap_hair	13	11.5	11	10	9.5	9	8.5	8
cap_stem	41	35	32	29	26.5	24	21.5	19
cap_curve	47	40	37	33	30	27	24	21
cap_ess	43	38	35	31	27	23	19	14
rule_thickness	.60	.44	.40	.38	.36	.34	.31	.28
dish	1	1	1	1	1	.9	.8	.7
bracket	34	24	20	18	16	14	12	10
jut	42	33	28	25	22	19	17	15
cap_jut	50	41	37	33	29	26	23	20
beak_jut	15	11.4	10	9.2	8.4	7.6	6.8	6
beak	121	84	70	63	56	49	42	35
vair	10	8.5	8	8	8	7	7	6
notch_cut	17	12	10	9	8	7	6	5
bar	16	12.5	11	10	9.5	9	8.5	8
slab	16	12.5	11	10	9.5	9	8.5	8
cap_bar	16	12.5	11	10	9.5	9	8.5	8
cap_band	16	12.5	11	10	9.5	9	8.5	8
cap_notch_cut	17	12	10	9	8	7	6	5
serif_drop	7	4.8	4	3.6	3.2	2.8	2.4	2
stem_corr	1	1	1	1	1	.9	.8	.7
vair_corr	1	1	1	1	1	.9	.7	.5
apex_corr	0	0	0	0	0	0	0	0
o	10	9	8	7	6	5	4.5	4
apex_o	10	9	8	7	6	5	4.5	4
slant	0	0	0	0	0	0	0	0
fudge	1	1	1	1	1	1	1	1
math_spread	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1
superness	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	1/6	1/6	1/6	1/6	1/6	1/6	1/6	1/6
beak_darkness	11/30	11/30	11/30	11/30	11/30	11/30	11/30	11/30
ligs	2	2	2	2	2	2	2	1
square_dots	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
hefty	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
low_asterisk	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`

All dimensions in this table represent multiples of $^1/_{36}$ pt, except in the cases of **font_size**, *rule_thickness*, *notch_cut*, and *cap_notch_cut*, which are given in pt. (The parameter values for *slant*, . . . , *ligs* are pure numbers, not dimensions.)

The table shows that most of the "vertical" dimensions in `cmr` fonts are scaled in a straightforward way; for example, the *x_height* of `cmr5` is exactly half the *x_height* of `cmr10`. But horizontally the smaller fonts are more extended, and they have relatively heavier weight (especially in hairlines).

Computer Modern Slanted Roman types appear in four standard sizes; the fonts are called `cmsl12`, `cmsl10`, `cmsl9`, and `cmsl8`. The four parameter files need not be shown here, since they are completely identical to their unslanted counterparts `cmr12`–`cmr8`, except that the **font_identifier** is `"CMSL"` and the *slant* parameter is 1/6.

Computer Modern Bold Extended Roman types appear in seven standard sizes, from 12 pt down to 5 pt; they are called `cmbx12`–`cmbx5`. Vertical dimensions of `cmbx` fonts are similar to the corresponding `cmr` parameters, except that the *x_height* is a bit taller and the *bar_height* is not quite so tall. Bold letters are generated with bolder pen strokes, of course, and the characters are rather more extended; for example, one "em" is 10 pt in `cmr10`, but 11.5 pt in `cmbx10`. A larger value of *superness* makes the bowls larger.

name	cmbx12	cmbx10	cmbx9	cmbx8	cmbx7	cmbx6	cmbx5
font_identifier	CMBX	CMBX	CMBX	CMBX	CMBX	CMBX	CMBX
font_size	12	10	9	8	7	6	5
u	27	23	21.3	19.6	17.9	16.2	14.5
width_adj	13	11	10.2	9.4	8.6	7.8	7
serif_fit	0	0	0	0	0	0	0
cap_serif_fit	8	7	6.2	5.4	4.6	3.8	3
letter_fit	0	0	0	0	1	2	3
body_height	324	270	243	216	189	162	135
asc_height	300	250	225	200	175	150	125
cap_height	296.4	247	222.3	197.6	172.9	148.2	123.5
fig_height	278.4	232	208.8	185.6	162.4	139.2	116
x_height	192	160	144	128	112	96	80
math_axis	108	90	81	72	63	54	45
bar_height	102	85	76	67	58	49	40
comma_depth	84	70	63	56	49	42	35
desc_depth	84	70	63	56	49	42	35
crisp	0	0	0	0	0	0	0
tiny	13	13	13	12	11	10	9
fine	10	10	10	10	9	9	8
thin_join	10.5	10	10	10	9	9	8
hair	18.3	17	16.6	16.2	15.8	15.4	15
stem	47	41	38	35	32	28	24
curve	52	46	43	39	35	31	27
ess	44	38	35	32	29	25	19
flare	51	45	42	39	35	31	27

dot_size	62	56	51	46	41	36	31
cap_hair	20	19	19	19	18	18	17
cap_stem	57	51	47	43	38	33	28
cap_curve	61	55	51	46	41	36	31
cap_ess	59	53	48	43	37	31	24
rule_thickness	.66	.60	.57	.54	.51	.47	.43
dish	1	1	1	1	.9	.8	.7
bracket	12	10	9	8	7	6	5
jut	30	25	22	19	17	15	13
cap_jut	46	39	35	31	27	23	19
beak_jut	12.4	11	10.2	9.4	8.6	7.8	7
beak	84	70	63	55	46	36	25
vair	14.3	13	12.5	12	11.5	11	10
notch_cut	12	10	9	8	7	6	5
bar	18.5	17	16	15	14	13	12
slab	18.5	17	16	15	14	13	12
cap_bar	18.5	17	16	15	14	13	12
cap_band	18.5	17	16	15	14	13	12
cap_notch_cut	12	10	9	8	7	6	5
serif_drop	3.6	3	2.7	2.4	2.1	1.8	1.5
stem_corr	2	2	1.9	1.8	1.7	1.6	1.5
vair_corr	1.5	1.5	1.4	1.3	1.2	1.1	1
apex_corr	0	0	0	0	2	5	9
o	5	4	4	4	3	3	3
apex_o	4	3	3	3	3	3	3
slant	0	0	0	0	0	0	0
fudge	1	1	1	1	1	1	1
math_spread	0.4	0.5	0.7	0.9	1.1	1.3	1.5
superness	8/11	8/11	8/11	8/11	8/11	8/11	8/11
superpull	1/8	1/8	1/8	1/8	1/8	1/8	1/8
beak_darkness	0.4	0.4	0.4	0.4	0.4	0.4	0.4
ligs	2	2	2	2	2	2	2
square_dots	**false**	**false**	**false**	**false**	**false**	**false**	**false**
hefty	**false**	**false**	**false**	**false**	**false**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**false**	**false**	**false**	**false**	**false**	**false**
low_asterisk	**false**	**false**	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`

Positive values of *apex_corr* are used at small sizes to keep diagonal junctions from filling in, even though these fonts are not '*hefty*'.

`Computer Modern Typewriter types` have four standard sizes, namely `cmtt12`, `cmtt10`, `cmtt9`, and `cmtt8`. These fonts have the same *x_height* values as their roman counterparts, but they have shorter ascenders and longer descenders. The stroke weights are almost all identical to the corresponding roman stem weight; there is essentially no contrast between thick and thin. Serifs are rounded and unbracketed. Furthermore, all characters have the same width.

Three standard variations of `cmtt10` are also provided, to demonstrate some of the other possibilities: `cmsltt10` is slanted; `cmvtt10` has characters of variable width; `cmtex10` has the TEX extended-ASCII character set.

Since typewriter fonts are quite different from the ordinary roman and bold fonts, the parameter values in the following table are particularly interesting. For example, one of the more subtle points is the unusually high setting of *math_axis*; this raises the mathematical signs so that they blend particularly well with uppercase letters and digits, as desired in applications to computer program listing. Such applications also account for *low_asterisk* being **true**.

name	`cmtt12`	`cmtt10`	`cmtt9`	`cmtt8`	`cmsltt10`	`cmvtt10`	`cmtex10`
font_identifier	CMTT	CMTT	CMTT	CMTT	CMSLTT	CMVTT	CMTEX
font_size	12	10	9	8	10	10	10
u	24.7	21	18.9	17	21	21	21
width_adj	0	0	0	0	0	0	0
serif_fit	0	0	0	0	0	0	0
cap_serif_fit	0	0	0	0	0	0	0
letter_fit	0	0	0	0	0	0	0
body_height	300	250	225	200	250	250	250
asc_height	264	220	198	176	220	220	220
cap_height	264	220	198	176	220	220	220
fig_height	264	220	198	176	220	220	220
x_height	186	155	139.5	124	155	155	155
math_axis	132	110	99	88	110	110	110
bar_height	95	79	71	63	79	79	79
comma_depth	60	50	45	40	50	50	50
desc_depth	96	80	72	64	80	80	80
crisp	25	22	21	19	22	22	22
tiny	25	22	21	19	22	22	22
fine	22	21	20	18	21	21	21
thin_join	22	21	20	18	21	21	21
hair	28	25	24	22	25	25	25
stem	28	25	24	22	25	25	25
curve	28	25	24	22	25	25	25
ess	25	22	21	19	22	22	22
flare	35	32	30	28	32	32	32

dot_size	39	36	33	31	36	36	36
cap_hair	28	25	24	22	25	25	25
cap_stem	28	25	24	22	25	25	25
cap_curve	28	25	24	22	25	25	25
cap_ess	28	25	24	22	25	25	25
rule_thickness	28/36	25/36	24/36	22/36	25/36	25/36	25/36
dish	0	0	0	0	0	0	0
bracket	0	0	0	0	0	0	0
jut	39	34	31	27	34	34	34
cap_jut	39	34	31	27	34	34	34
beak_jut	0	0	0	0	0	0	0
beak	39	34	31	27	34	34	34
vair	25	22	21	19	22	22	22
notch_cut	28/36	25/36	24/36	22/36	25/36	25/36	25/36
bar	25	22	21	19	22	22	22
slab	25	22	21	19	22	22	22
cap_bar	25	22	21	19	22	22	22
cap_band	25	22	21	19	22	22	22
cap_notch_cut	28/36	25/36	24/36	22/36	25/36	25/36	25/36
serif_drop	0	0	0	0	0	0	0
stem_corr	0	0	0	0	0	0	0
vair_corr	0	0	0	0	0	0	0
apex_corr	11	10	9	8	10	10	10
o	5	4	3.5	3	4	4	4
apex_o	4	3	3	3	3	3	3
slant	0	0	0	0	1/6	0	1/4
fudge	0.86	0.81	0.81	0.81	0.81	0.81	0.81
math_spread	−1	−1	−1	−1	−1	−1	−1
superness	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	0	0	0	0	0	0	0
beak_darkness	0	0	0	0	0	0	0
ligs	0	0	0	0	0	2	0
square_dots	**false**	**false**	**false**	**false**	**false**	**false**	**false**
hefty	**true**	**true**	**true**	**true**	**true**	**true**	**true**
serifs	**true**	**true**	**true**	**true**	**true**	**true**	**true**
monospace	**true**	**true**	**true**	**true**	**true**	**false**	**true**
variant_g	**false**	**false**	**false**	**false**	**false**	**false**	**false**
low_asterisk	**true**	**true**	**true**	**true**	**true**	**true**	**true**
math_fitting	**false**	**false**	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`	`texset`

Two additional standard fonts, called `cmtex9` and `cmtex8`, are created by modifying `cmtt9` and `cmtt8`, just as `cmtt10` has been changed to make `cmtex10`.

Computer Modern Sans Serif types come in five standard sizes, from 17 pt to 8 pt; the fonts are called `cmss17`, `cmss12`, `cmss10`, `cmss9`, and `cmss8`. The parameter settings are analogous in many respects to the corresponding `cmr` fonts, with which they are supposed to be compatible. For example, the values of u, *asc_height*, *desc_depth* are identical in seriffed and unseriffed fonts.

The main difference between `cmss10` and `cmr10`, besides the fact that `cmss10` has *serifs* = **false**, is the "monoline" nature of `cmss10`. There is almost no contrast between thick and thin strokes, because *hair*, *stem*, and *curve* are all equal to a value that falls between the *stem* and *curve* specifications of a seriffed font. In a monoline font, *hefty* is made **true**, and *notch_cut* is used to reduce heavy junction points.

Computer Modern Slanted Sans Serif types come in the same five sizes. Their names (`cmssi17`, `cmssi12`, `cmssi10`, `cmssi9`, and `cmssi8`) are slightly misleading, because the '`i`' does not stand for 'italic'; the character shapes follow the roman style. All parameters are the same in corresponding members of the `"CMSS"` and `"CMSSI"` families, except that the *slant* values in the latter case are set to produce an angle of 12°.

name	`cmss17`	`cmss12`	`cmss10`	`cmss9`	`cmss8`	`cmssi10`
font_identifier	`CMSS`	`CMSS`	`CMSS`	`CMSS`	`CMSS`	`CMSSI`
font_size	17.28	12	10	9	8	10
u	32.5	23.5	20	18.5	17	20
width_adj	24	17	15	13	12	15
serif_fit	−12	−9	−7	−6.5	−6	−7
cap_serif_fit	−17	−13	−10	−9	−8.5	−10
letter_fit	0	0	0	0	0	0
body_height	466	324	270	243	216	270
asc_height	432	300	250	225	200	250
cap_height	432	300	250	225	200	250
fig_height	408	283	236	212	189	236
x_height	267.8	192	160	144	128	160
math_axis	155.5	108	90	81	72	90
bar_height	136	97	79	71	63	79
comma_depth	78	54	45	41	37	45
desc_depth	121	84	70	63	56	70
crisp	0	0	0	0	0	0
tiny	0	0	0	0	0	0
fine	0	0	0	0	0	0
thin_join	28	20	17	16	15	17
hair	47	33	28	27	25	28
stem	47	33	28	27	25	28
curve	47	33	28	27	25	28
ess	47	33	28	27	25	28
flare	42	30	25	24	22	25

dot_size	50	35	30	28	26	30
cap_hair	47	33	28	27	25	28
cap_stem	52	37	32	31	29	32
cap_curve	52	37	32	31	29	32
cap_ess	52	37	32	31	28	32
rule_thickness	.60	.44	.40	.38	.36	.40
dish	0	0	0	0	0	0
bracket	0	0	0	0	0	0
jut	0	0	0	0	0	0
cap_jut	0	0	0	0	0	0
beak_jut	0	0	0	0	0	0
beak	.8	.6	.5	.5	.4	.5
vair	38	26	22	21	19	22
notch_cut	32/36	22/36	18/36	17/36	16/36	18/36
bar	36	25	21	20	18	21
slab	40	27	23	22	20	23
cap_bar	38	26	22	21	19	22
cap_band	35	24	20	19	17	20
cap_notch_cut	46/36	31/36	25/36	24/36	22/36	25/36
serif_drop	2	2	2	2	2	2
stem_corr	1	1	1	1	1	1
vair_corr	1	1	1	1	1	1
apex_corr	7	6	5	4.5	4	5
o	10	9	8	7	6	8
apex_o	0	0	0	0	0	0
slant	0	0	0	0	0	sind 12/cosd 12
fudge	1	.98	.96	.95	.93	.96
math_spread	-0.4	-0.2	0	0.2	0.4	0
superness	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	1/20	1/20	1/20	1/20	1/20	1/20
beak_darkness	0	0	0	0	0	0
ligs	2	2	2	2	2	2
square_dots	**true**	**true**	**true**	**true**	**true**	**true**
hefty	**true**	**true**	**true**	**true**	**true**	**true**
serifs	**false**	**false**	**false**	**false**	**false**	**false**
monospace	**false**	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**false**	**false**	**false**	**false**	**false**
low_asterisk	**false**	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`	`roman`

The parameter settings for `cmssi17`, `cmssi12`, `cmssi9`, and `cmssi8` are completely analogous to those shown for `cmssi10`.

Computer Modern Sans Serif Demibold Condensed type, also called `cmssdc10`, is one of several standard fonts that demonstrate additional possibilities for sans-serif typefaces in the Computer Modern family. The chapter titles in this book (and in the entire *Computers and Typesetting* series) have been set in '`cmssdc10 at 40pt`', a four-fold magnification of the 10 pt size.

Computer Modern Sans Serif Bold Extended type, also known as `cmssbx10`, is another such "demonstration" font. The big numbers and letters that introduce chapters and appendices in *The TEXbook* and *The METAFONTbook* have been set in a special version of `cmssbx10` called `cminch`; this name reflects the fact that the letters and digits are exactly an inch high. (The parameters for `cminch` are defined just like those for `cmssbx10`, except that `cminch.mf` begins with the line '**numeric** $Pt\#$; $^{250}/_{36}Pt\# = 1\ \text{in}\#$'; then Pt is used in place of pt in all the parameter assignments.)

Computer Modern Sans Serif Quotation Style type demonstrates yet another possible variation, used in the quotations and epigraphs of this book. The standard font `cmssq8` has the body size of eight-point type, but its *x_height* is almost the *x_height* of `cmr10`. There is a slanted version called `cmssqi8`.

name	cmssdc10	cmssbx10	cmss10	cmssqi8	cmssq8
font_identifier	CMSSDC	CMSSBX	CMSS	CMSSQI	CMSSQ
font_size	10	10	10	8	8
u	19	22	20	20	20
width_adj	15	22	15	2	2
serif_fit	−8	−9	−7	−10	−10
cap_serif_fit	−9	−11	−10	−14	−14
letter_fit	0	0	0	0	0
body_height	270	270	270	220	220
asc_height	250	250	250	200	200
cap_height	250	250	250	197	197
fig_height	250	250	236	190	190
x_height	170	165	160	150	150
math_axis	95	90	90	80	80
bar_height	95	90	79	80	80
comma_depth	40	38	45	40	40
desc_depth	60	70	70	40	40
crisp	23	26	0	18	18
tiny	23	26	0	18	18
fine	8	9	0	8	8
thin_join	8	9	17	8	8
hair	40	49	28	23	23
stem	40	49	28	23	23
curve	40	49	28	23	23
ess	37	43	28	23	23
flare	31	40	25	21	21

dot_size	38	47	30	28	28
cap_hair	40	49	28	23	23
cap_stem	44	53	32	27	27
cap_curve	44	53	32	27	27
cap_ess	42	51	32	27	27
rule_thickness	25/36	30/36	.40	20/36	20/36
dish	0	0	0	0	0
bracket	0	0	0	0	0
jut	0	0	0	0	0
cap_jut	0	0	0	0	0
beak_jut	0	0	0	0	0
beak	1.5	1.5	.5	0	0
vair	23	26	22	18	18
notch_cut	40/36	49/36	18/36	23/26	23/26
bar	23	26	21	18	18
slab	31	36	23	19	19
cap_bar	29	34	22	19	19
cap_band	27	31	20	19	19
cap_notch_cut	37/36	45/36	25/36	23/26	23/26
serif_drop	2	2	2	0	0
stem_corr	1	1	1	.5	.5
vair_corr	2	2	1	.5	.5
apex_corr	14	18	5	4	4
o	7	8	8	6	6
apex_o	0	0	0	2	2
slant	0	0	0	sind 12/cosd 12	0
fudge	.91	.88	.96	.95	.95
math_spread	0.3	0.5	0	0	0
superness	0.74	0.75	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	1/18	1/18	1/20	1/10	1/10
beak_darkness	0	0	0	0	0
ligs	2	2	2	2	2
square_dots	**true**	**true**	**true**	**true**	**true**
hefty	**true**	**true**	**true**	**true**	**true**
serifs	**false**	**false**	**false**	**false**	**false**
monospace	**false**	**false**	**false**	**false**	**false**
variant_g	**true**	**false**	**false**	**true**	**true**
low_asterisk	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`

The parameters of `cmss10` have been repeated here for comparison. The new fonts have "softer" edges, because *crisp*, *tiny*, and *fine* are positive.

Computer Modern Dunhill Roman type demonstrates yet another kind of parameter variation. Font `cmdunh10` has the standard parameters of `cmr10`, except that *body_height*, *asc_height*, *cap_height*, and *fig_height* have been drastically increased, by $^{100}/_{36}$ pt each.

Computer Modern Bold Extended Slanted Roman type, also known as `cmbxsl10`, is a slanted companion to `cmbx10`. There's also a standard font called `cmb10`, which is a non-extended bold roman. Characters have almost the same widths in `cmb10` and `cmr10`, because both fonts have $u = {}^{20}/_{36}$ pt.

Computer Modern Funny Roman type carries the idea of parameter variation to a ridiculous extreme. The author concocted it primarily to test the robustness of his programs, but people have also used `cmff10` to typeset party invitations. Its backward slant, large *x_height* and *desc_depth*, small *bar_height*, twiggy stems, diamond-shaped bowls, reverse beaks, and deep notches combine to make rather bizarre effects. Proofmode-size blowups of these freakish characters have a curious property: They convey much of a design's "meta-ness" in a single drawing. More normal settings of the parameters do not do this as well.

name	`cmdunh10`	`cmbxsl10`	`cmb10`	`cmff10`	`cmfib8`
font_identifier	`CMDUNH`	`CMBXSL`	`CMB`	`CMFF`	`CMFIB`
font_size	10	10	10	10	8
u	20	23	20	16	21
width_adj	0	11	10	−16	0
serif_fit	0	0	0	−3	−2
cap_serif_fit	5	7	6	−4	0
letter_fit	0	0	0	2	0
body_height	370	270	270	240	233
asc_height	350	250	250	220	233
cap_height	346	247	246	225	233
fig_height	332	232	232	210	233
x_height	155	160	160	190	144
math_axis	90	90	90	55	89
bar_height	87	87	85	60	89
comma_depth	70	70	70	70	55
desc_depth	70	70	70	100	55
crisp	0	0	0	10	0
tiny	8	8	13	0	13
fine	7	10	10	0	13
thin_join	7	10	10	14	13
hair	9	17	17	12	21
stem	25	41	41	16	34
curve	30	46	46	20	34 + 5
ess	27	38	38	22	34
flare	33	45	45	24	34

dot_size	38	56	56	26	34
cap_hair	11	19	19	16	21
cap_stem	32	51	51	24	43
cap_curve	37	55	55	26	$34+5$
cap_ess	35	53	53	28	34
rule_thickness	.40	.60	.60	.50	$21/36$
dish	1	1	1	5	1
bracket	20	10	10	20	21
jut	28	25	25	3	21
cap_jut	37	39	34	4	21
beak_jut	10	11	11	-8	13
beak	70	70	70	10	55
vair	8	13	13	14	13
notch_cut	10	10	10	$2/36$	8
bar	11	17	17	17	13
slab	11	17	17	20	13
cap_bar	11	17	17	22	13
cap_band	11	17	17	17	13
cap_notch_cut	10	10	10	$3/36$	8
serif_drop	4	3	3	-10	3
stem_corr	1	2	2	-1	1
vair_corr	1	1.5	1.5	-1	1
apex_corr	0	0	0	6	0
o	8	4	4	4	8
apex_o	8	3	3	9	8
slant	0	$1/6$	0	$-.1$	0
fudge	1	1	1	1	1
math_spread	0	0.5	0.5	0.5	$3/8$
superness	$1/\sqrt{2}$	$8/11$	$8/11$	$2/3$	$1/\sqrt{2}$
superpull	$1/6$	$1/8$	$1/8$	$-1/8$	$1/13$
beak_darkness	$11/30$	0.4	0.4	0.5	$5/13$
ligs	2	2	2	2	2
square_dots	**false**	**false**	**false**	**true**	**false**
hefty	**false**	**false**	**false**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**false**	**false**	**false**	**false**
low_asterisk	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`roman`	`roman`	`roman`	`roman`

The fifth column, `cmfib8`, generates a standard font based on the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .), in hopes that these numbers will yield an especially pleasing alphabet.

Computer Modern Text Italic types come in five standard sizes: `cmti12`, `cmti10`, `cmti9`, `cmti8`, and `cmti7`. Uppercase italic letters are generated by the same programs that make uppercase roman letters, but the lowercase letters belong to an entirely different style. Some of the numerals (*1, 2, 3, 4, 7*) are also changed, as are special symbols like question mark and ampersand (*?, &*). The `textit` driver program controls such stylistic variations.

It is interesting to compare the parameters of `cmti12` to those of `cmr12`, and to compare `cmti10` to `cmr10`, etc.; so `cmr12` is repeated in the table below. Text italic letters are slightly condensed (because they have a smaller value of u); their stems are slightly lighter, and this makes the total effect lighter even though the hairlines have gained some weight. A significant *letter_fit* is added at the left and right of each character, to keep the letters from bunching up. The slant ratio is 1/4 (compared to 1/6 in slanted roman fonts). The *crisp* parameter is positive, hence text italic serifs have softer edges. These differences all cause *UPPERCASE TEXT ITALIC LETTERS* to be different from *UPPERCASE SLANTED ROMAN LETTERS*.

name	`cmr12`	`cmti12`	`cmti10`	`cmti9`	`cmti8`	`cmti7`
font_identifier	CMR	CMTI	CMTI	CMTI	CMTI	CMTI
font_size	12	12	10	9	8	7
u	23.5	21.6	18.4	17	15.8	14.8
width_adj	0	0	0	0	0	0
serif_fit	0	0	0	0	0	0
cap_serif_fit	5.8	5.8	5	4.6	4.1	3.5
letter_fit	0	10.8	9.2	8.5	7.9	9.4
body_height	324	324	270	243	216	189
asc_height	300	300	250	225	200	175
cap_height	295.2	295.2	246	221.4	196.8	172.2
fig_height	278.4	278.4	232	208.8	185.6	162.4
x_height	186	186	155	139.5	124	108.5
math_axis	108	108	90	81	72	63
bar_height	104.4	104.4	87	78.3	69.6	60.9
comma_depth	84	84	70	63	56	49
desc_depth	84	84	70	63	56	49
crisp	0	8	8	8	8	7
tiny	8	8	8	8	8	7
fine	7	7	7	7	7	6
thin_join	7.5	7.5	7	7	7	6
hair	9.5	12	11	11	11	10.5
stem	28	26	23	22	20	18.5
curve	33	31	28	26	23.5	21.5
ess	30	29	27	25	23	21
flare	36	32	29	26	24	21

dot_size	42	42	38	35	32	29
cap_hair	11.5	11.5	11	11	11	10.5
cap_stem	35	32	29	26	23.5	21.5
cap_curve	40	37	34	30	27	24.5
cap_ess	38	34	31	27	24	21
rule_thickness	.44	.44	.40	.38	.36	.34
dish	1	1	1	1	1	.9
bracket	24	24	20	18	16	14
jut	33	35	30	27	24	21
cap_jut	41	38	34	30.5	27	24
beak_jut	11.4	10.4	9	8.5	7.9	7.4
beak	84	84	70	63	56	49
vair	8.5	8.5	8	8	8	7
notch_cut	12	12	10	9	8	7
bar	12.5	12.5	11	10	9.5	9
slab	12.5	12.5	11	10	9.5	9
cap_bar	12.5	12.5	11	10	9.5	9
cap_band	12.5	12.5	11	10	9.5	9
cap_notch_cut	12	12	10	9	8	7
serif_drop	4.8	4.8	4	3.6	3.2	2.8
stem_corr	1	1	1	1	1	.9
vair_corr	1	1	1	1	1	.9
apex_corr	0	0	0	0	0	0
o	9	9	8	7	6	5
apex_o	9	9	8	7	6	5
slant	0	1/4	1/4	1/4	1/4	1/4
fudge	1	1	1	1	1	1
math_spread	−0.2	−0.2	0	0.2	0.4	0.6
superness	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	1/6	1/6	1/6	1/6	1/6	1/6
beak_darkness	11/30	11/30	11/30	11/30	11/30	11/30
ligs	2	2	2	2	2	2
square_dots	**false**	**false**	**false**	**false**	**false**	**false**
hefty	**false**	**false**	**false**	**false**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**true**	**true**	**true**	**true**	**true**
low_asterisk	**false**	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**false**	**false**	**false**	**false**	**false**
generate	`roman`	`textit`	`textit`	`textit`	`textit`	`textit`

The *variant_g* values are **true**, since italic g's are *g*'s; but this parameter is actually irrelevant in fonts generated by `textit`.

Computer Modern Math Italic types have seven standard sizes, from 12 pt down to 5 pt; the font names are `cmmi12`–`cmmi5`. In this case *math_fitting* is **true**, so the characters are positioned in a special way to enhance their appearance in mathematical formulas. Furthermore, a math italic font has a completely different character set from text italic: The lowercase Greek alphabet is included, and there are special symbols like Weierstrass's 'p' ($\wp$). The `mathit` driver program controls such stylistic variations.

Many of the characters of math italic fonts are unslanted, as if the *slant* parameter had been zero. This includes the numerals, which are in "old style" (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), as well as more than a dozen special symbols like '$\angle$', '$\triangleright$', '$\flat$', '$\natural$', '$\sharp$', and '$\star$'. Appendix F of *The TEXbook* observes that "this portion of the font doesn't deserve the name math italic; it's really a resting place for characters that don't fit anywhere else."

It is interesting to compare the parameters of `cmmi12` to those of `cmr12`, and to compare `cmmi10` to `cmr10`, etc.; so `cmr12` is repeated in the table below. Math italic letters are not condensed as text italic was; and they are not quite as light as text italic. (After all, mathematics is important to mathematicians.)

name	`cmr12`	`cmmi12`	`cmmi10`	`cmmi9`	`cmmi8`	`cmmi7`	`cmmi6`	`cmmi5`
font_identifier	CMR	CMMI	CMMI	CMMI	CMMI	CMMI	CMMI	CMMI
font_size	12	12	10	9	8	7	6	5
u	23.5	23.5	20	18.5	17	15.5	14	12.5
width_adj	0	0	0	0	0	0	0	0
serif_fit	0	0	0	0	0	0	0	0
cap_serif_fit	5.8	5.8	5	4.6	4.1	3.5	2.8	2
letter_fit	0	0	0	0	0	4	6	10
body_height	324	324	270	243	216	189	162	135
asc_height	300	300	250	225	200	175	150	125
cap_height	295.2	295.2	246	221.4	196.8	172.2	147.6	123
fig_height	278.4	278.4	232	208.8	185.8	162.4	139.2	116
x_height	186	186	155	139.5	124	108.5	93	77.5
math_axis	108	108	90	81	72	63	54	45
bar_height	104.4	104.4	87	78.3	69.6	60.9	52.2	43.5
comma_depth	84	84	70	63	56	49	42	35
desc_depth	84	84	70	63	56	49	42	35
crisp	0	8	8	8	8	7	6.5	6
tiny	8	8	8	8	8	7	6.5	6
fine	7	7	7	7	7	6	6	5
thin_join	7.5	7.5	7.5	7	7	6	6	5
hair	9.5	9.5	9	9	9	8.5	8	7.5
stem	28	27	24	23	21.5	19.5	17.5	15.5
curve	33	32	29	27	25	22.5	20	17.5
ess	30	30	27	25	23	21	17	12
flare	36	35	32	29	26.5	23.5	20.5	18.5

dot_size	42	42	38	35	32	29	26	22
cap_hair	11.5	11.5	11	10	9.5	9	8.5	8
cap_stem	35	33	30	28	25.5	23.5	21	18.5
cap_curve	40	38	35	32	29	26.5	23.5	20.5
cap_ess	38	32	33	30	26	22.5	18.5	13.5
rule_thickness	.44	.44	.40	.38	.36	.34	.31	.28
dish	1	1	1	1	1	.9	.8	.7
bracket	24	24	20	18	16	14	12	10
jut	33	35	30	27	24	21	19	17
cap_jut	41	41	37	33	29	26	23	20
beak_jut	11.4	11.4	10	9.2	8.4	7.6	6.8	6
beak	84	84	70	63	56	49	42	35
vair	8.5	8.5	8	8	8	7	7	6
notch_cut	12	12	10	9	8	7	6	5
bar	12.5	12.5	11	10	9.5	9	8.5	8
slab	12.5	12.5	11	10	9.5	9	8.5	8
cap_bar	12.5	12.5	11	10	9.5	9	8.5	8
cap_band	12.5	12.5	11	10	9.5	9	8.5	8
cap_notch_cut	12	12	10	9	8	7	6	5
serif_drop	4.8	4.8	4	3.6	3.2	2.8	2.4	2
stem_corr	1	1	1	1	1	.9	.8	.7
vair_corr	1	1	1	1	1	.9	.7	.5
apex_corr	0	0	0	0	0	0	0	0
o	9	9	8	7	6	5	4.5	4
apex_o	9	9	8	7	6	5	4.5	4
slant	0	1/4	1/4	1/4	1/4	1/4	1/4	1/4
fudge	1	1	1	1	1	1	1	1
math_spread	−0.2	−0.2	0	0.2	0.4	0.6	0.8	1
superness	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$	$1/\sqrt{2}$
superpull	1/6	1/6	1/6	1/6	1/6	1/6	1/6	1/6
beak_darkness	11/30	11/30	11/30	11/30	11/30	11/30	11/30	11/30
ligs	2	2	2	2	2	2	2	1
square_dots	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
hefty	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
variant_g	**false**	**true**	**true**	**true**	**true**	**true**	**true**	**true**
low_asterisk	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**
math_fitting	**false**	**true**	**true**	**true**	**true**	**true**	**true**	**true**
generate	`roman`	`mathit`	`mathit`	`mathit`	`mathit`	`mathit`	`mathit`	`mathit`

The *variant_g* values are **true**, since math g's are *g*'s; but the `mathit` driver program does not actually look at the parameters *variant_g* or *low_asterisk*.

Computer Modern Bold Extended Text Italic type, also known as `cmbxti10`, is one of several standard fonts that demonstrate additional possibilities for italic typefaces in the Computer Modern family. There's also a bold math italic, called `cmmib10`. (Standard font names have been chosen in such a way that there are no conflicts when names longer than six characters are shortened by retaining the first three and last three characters. For example, the file names `cmbxti10.mf` and `cmmib10.mf` can safely become '`cmbi10.mf`' and '`cmmb10.mf`' on computer systems that do not allow long file names.)

Computer Modern Italic Typewriter type, also known as `cmitt10`, is another such "demonstration" font. It is a monospaced text italic, with weights and spacing compatible with `cmtt10` and `cmsltt10`.

Computer Modern Unslanted Text Italic type, `cmu10`, demonstrates yet another possible variation. It has italic-style letters, but they are upright and they have the weights of `cmr10`.

Finally, there's also *Computer Modern Funny Italic* (`cmfi10`), a companion to the funny font `cmff10`. This one is not quite as hilarious as the other.

name	cmbxti10	cmmib10	cmitt10	cmu10	cmfi10
font_identifier	CMBXTI	CMMIB	CMITT	CMU	CMFI
font_size	10	10	10	10	10
u	21.2	23	21	20	21
width_adj	11	11	0	0	−16
serif_fit	0	0	0	0	−3
cap_serif_fit	7	7	0	5	−4
letter_fit	11	0	15	10	2
body_height	270	270	250	270	240
asc_height	250	250	220	250	220
cap_height	247	247	220	247	225
fig_height	232	232	220	232	210
x_height	160	160	155	155	190
math_axis	90	90	110	90	55
bar_height	85	85	79	87	60
comma_depth	70	70	50	70	70
desc_depth	70	70	80	70	100
crisp	13	13	22	0	11
tiny	13	13	22	8	0
fine	10	10	21	7	0
thin_join	10	10	21	7	14
hair	20	17	25	9	11
stem	38	40	25	25	17
curve	43	45	25	30	21
ess	35	36	22	22	23
flare	42	42	32	33	25

dot_size	53	56	36	38	26
cap_hair	22	19	25	11	15
cap_stem	49	50	25	32	25
cap_curve	52	53	25	37	27
cap_ess	50	51	25	32	29
rule_thickness	.60	.60	25/36	.40	.50
dish	1	1	0	1	5
bracket	10	10	0	20	20
jut	27	27	40	28	3
cap_jut	39	39	34	37	4
beak_jut	10	11	0	10	−8
beak	70	70	34	70	10
vair	13	13	22	8	15
notch_cut	10	10	25/36	10	2
bar	17	17	22	11	18
slab	17	17	22	11	21
cap_bar	17	17	22	11	23
cap_band	17	17	22	11	18
cap_notch_cut	10	10	25/36	10	3/36
serif_drop	3	3	0	4	−10
stem_corr	2	2	0	1	−1
vair_corr	1.5	1.5	1.5	1	−1
apex_corr	0	0	10	0	6
o	6	6	4	8	4
apex_o	6	6	3	8	9
slant	1/4	1/4	1/4	0	+.1
fudge	1	1	.81	1	1
math_spread	0.5	0.5	−1	0	0.5
superness	8/11	8/11	$1/\sqrt{2}$	$1/\sqrt{2}$	2/3
superpull	1/8	1/8	0	1/6	−1/8
beak_darkness	0.4	0.4	0	11/30	0.5
ligs	2	2	0	2	2
square_dots	**false**	**false**	**false**	**false**	**true**
hefty	**false**	**false**	**true**	**false**	**false**
serifs	**true**	**true**	**true**	**true**	**true**
monospace	**false**	**false**	**true**	**false**	**false**
variant_g	**true**	**true**	**true**	**true**	**true**
low_asterisk	**false**	**false**	**true**	**false**	**false**
math_fitting	**false**	**true**	**false**	**false**	**false**
generate	`textit`	`mathit`	`textit`	`textit`	`textit`

A great many other possibilities for italic parameter settings can, of course, be imagined. For example, a slightly expanded demibold font with a *slant* of 1/8 seems to work well.

Computer Modern Caps and Small Caps types need a different sort of parameter file, because the "lowercase" letters are generated by uppercase programs; two different sets of uppercase parameters are needed.

The problem is solved by giving values to 62 parameters as usual, then giving additional values to 25 parameters with '*lower*' prefixed to their names. For example, *lower.u* is the unit width to use when generating uppercase letters in lowercase positions, and *lower.cap_height* is the height of the small caps. Some of the original 62 parameters apply only to lowercase letters, so they aren't actually used by the `csc` driver program; but values are given to all parameters, whether they are necessary or not.

There are two standard fonts to demonstrate the possibilities: `cmcsc10` is intended to mix well with `cmr10`, and `cmtcsc10` is a "Typewriter Caps and Small Caps" font analogous to `cmtt10`. Many more variations (bold and/or slanted and/or sans-serif) could be tried. The *letter_fit* and *lower.letter_fit* parameters seem to deserve special care.

name	`cmcsc10`	*lower*	`cmtcsc10`	*lower*
font_identifier	CMCSC		CMTCSC	
font_size	10		10	
u	21	16	21	21
width_adj	0	0	0	0
serif_fit	0		0	
cap_serif_fit	5	4	0	0
letter_fit	5	2.4	0	3
body_height	270	200	250	190
asc_height	250		220	
cap_height	246	185	220	170
fig_height	232		220	
x_height	155	116	155	120
math_axis	90		110	
bar_height	87	65	79	62
comma_depth	70	52	50	38
desc_depth	70		80	
crisp	0		22	
tiny	8		22	
fine	7		21	
thin_join	7		21	
hair	9		25	
stem	25	21	25	25
curve	30		25	
ess	27		22	
flare	33	25	32	30

dot_size	38		36	
cap_hair	11	9.1	25	25
cap_stem	32	26	25	25
cap_curve	37	31	25	25
cap_ess	35	28	25	25
rule_thickness	.40		25/36	
dish	1		0	
bracket	20		0	
jut	28		34	
cap_jut	37	26	34	26
beak_jut	10	7	0	0
beak	70	52	34	26
vair	8		22	
notch_cut	10		25/36	
bar	11		22	
slab	11	9.1	22	22
cap_bar	11	9.1	22	22
cap_band	11	9.1	22	22
cap_notch_cut	10	10	25/36	24/36
serif_drop	4		0	
stem_corr	1		0	
vair_corr	1		0	
apex_corr	0		10	
o	8	5	4	3
apex_o	8	4	3	2
slant	0		0	
fudge	1	1	0.81	0.85
math_spread	0		-1	
superness	$1/\sqrt{2}$		$1/\sqrt{2}$	
superpull	1/6		0	
beak_darkness	11/30		0	
ligs	1		0	
square_dots	**false**		**false**	
hefty	**false**		**true**	
serifs	**true**		**true**	
monospace	**false**		**true**	
variant_g	**false**		**false**	
low_asterisk	**false**		**true**	
math_fitting	**false**		**false**	
generate	`csc`		`csc`	

For example, we have *beak* $= {}^{70}/_{36}$ pt and *lower.beak* $= {}^{52}/_{36}$ pt in font `cmcsc10`. The `csc` driver file sets *beak* := *lower.beak* before generating the small caps.

The Computer Modern Math Symbol fonts contain 128 characters for use in math formulas. They come in six standard sizes, called `cmsy10`–`cmsy5`, and their 62 parameters are identical to the 62 parameters of the corresponding math italic fonts, `cmmi10`–`cmmi5`.

Besides the 62 parameters needed for font generation, the `cmsy` parameter files also supply 15 '**fontdimen**' parameters that control the way TEX will typeset with these fonts, as explained in *The TEXbook*.

For example, parameter file `cmsy10.mf` begins almost like `cmmi10.mf`, except that it specifies `"CMSY"` as the **font_identifier**. But where `cmmi10.mf` ends with '**generate** `mathit`', `cmsy10.mf` has the following mumbo jumbo:

```
% the following font parameters are explained in The TEXbook, Appendix G
subs := 7/10;                          % subscripts to 10pt will normally be in 7pt type
rth# := .4pt#;                         % assume that rules will come from cmex10
fontdimen 8:                           % parameters σ8 through σ22 will now be listed
  math_axis# + 3.51rth# + 54/36pt# + subs * desc_depth#,          % num1
  math_axis# + 1.51rth# + 30/36pt#,                               % num2
  math_axis# + 1.51rth# + 48/36pt#,                               % num3
  − (math_axis# − 3.51rth# − subs * fig_height# − 124/36pt#),     % denom1
  − (math_axis# − 1.51rth# − subs * fig_height# − 30/36pt#),      % denom2
  8.99pt# − subs * asc_height#,                                   % sup1
  8.49pt# − subs * asc_height#,                                   % sup2
  104/36pt#,                                                      % sup3
  54/36pt#, −(8.49pt# − 2subs * asc_height# − 3.1rth#),           % sub1, sub2
  subs * asc_height# − 36/36pt#, 18/36pt#,                        % sup_drop, sub_drop
  23.9pt#, 10.1pt#,                                               % delim1, delim2
  math_axis#;                                                     % axis_height
generate mathsy                        % switch to the driver file
```

Similarly, `cmsy9.mf` is like `cmmi9.mf` except that it ends with

```
subs := 6/9;                           % subscripts to 9pt will normally be in 6pt type
rth# := .4pt#;                         % assume that rules will come from cmex10
fontdimen 8:                           % parameters σ8 through σ22 will now be listed
  math_axis# + 3.51rth# + 36/36pt# + subs * desc_depth#,          % num1
  math_axis# + 1.51rth# + 30/36pt#,                               % num2
  math_axis# + 1.51rth# + 43/36pt#,                               % num3
  − (math_axis# − 3.51rth# − subs * fig_height# − 111/36pt#),     % denom1
  − (math_axis# − 1.51rth# − subs * fig_height# − 30/36pt#),      % denom2
  8.49pt# − subs * asc_height#,                                   % sup1
  7.99pt# − subs * asc_height#,                                   % sup2
  93/36pt#,                                                       % sup3
  36/36pt#, −(7.99pt# − 2subs * asc_height# − 3.1rth#),           % sub1, sub2
  subs * asc_height# − 36/36pt#, 18/36pt#,                        % sup_drop, sub_drop
  23.9pt#, 9.1pt#,                                                % delim1, delim2
```

after which comes '*math_axis#*; **generate** `mathsy`' (which we won't bother to mention any more because all symbol font parameter files end this way).

The corresponding lines of `cmsy8.mf` are:

$subs := {}^{6}/_{8}$;	% subscripts to 8pt will normally be in 6pt type
$rth\# := .4pt\#$;	% assume that rules will come from `cmex10`
fontdimen 8:	% parameters σ_8 through σ_{22} will now be listed
$math_axis\# + 3.51rth\# + {}^{36}/_{36}pt\# + subs * desc_depth\#$,	% num1
$math_axis\# + 1.51rth\# + {}^{24}/_{36}pt\#$,	% num2
$math_axis\# + 1.51rth\# + {}^{38}/_{36}pt\#$,	% num3
$-(math_axis\# - 3.51rth\# - subs * fig_height\# - {}^{98}/_{36}pt\#)$,	% denom1
$-(math_axis\# - 1.51rth\# - subs * fig_height\# - {}^{24}/_{36}pt\#)$,	% denom2
$7.49pt\# - subs * asc_height\#$,	% sup1
$6.99pt\# - subs * asc_height\#$,	% sup2
${}^{82}/_{36}pt\#$,	% sup3
${}^{36}/_{36}pt\#$, ${}^{72}/_{36}pt\#$,	% sub1, sub2
$subs * asc_height\# - {}^{36}/_{36}pt\#$, ${}^{18}/_{36}pt\#$,	% sup_drop, sub_drop
$11.9pt\#$, $9.1pt\#$,	% delim1, delim2

And for `cmsy7.mf` they are:

$subs := {}^{5}/_{7}$;	% subscripts to 7pt will normally be in 5pt type
$rth\# := .4pt\#$;	% assume that rules will come from `cmex10`
fontdimen 8:	% parameters σ_8 through σ_{22} will now be listed
$math_axis\# + 3.51rth\# + {}^{36}/_{36}pt\# + subs * desc_depth\#$,	% num1
$math_axis\# + 1.51rth\# + {}^{12}/_{36}pt\#$,	% num2
$math_axis\# + 1.51rth\# + {}^{34}/_{36}pt\#$,	% num3
$-(math_axis\# - 3.51rth\# - subs * fig_height\# - {}^{86}/_{36}pt\#)$,	% denom1
$-(math_axis\# - 1.51rth\# - subs * fig_height\# - {}^{12}/_{36}pt\#)$,	% denom2
$6.99pt\# - subs * asc_height\#$,	% sup1
$6.49pt\# - subs * asc_height\#$,	% sup2
${}^{72}/_{36}pt\#$,	% sup3
${}^{36}/_{36}pt\#$, ${}^{72}/_{36}pt\#$,	% sub1, sub2
$subs * asc_height\# - {}^{36}/_{36}pt\#$, ${}^{18}/_{36}pt\#$,	% sup_drop, sub_drop
$11.9pt\#$, $8.1pt\#$,	% delim1, delim2

And for `cmsy6.mf`:

$subs := {}^{5}/_{6}$;	% subscripts to 6pt will normally be in 5pt type
$rth\# := .4pt\#$;	% assume that rules will come from `cmex10`
fontdimen 8:	% parameters σ_8 through σ_{22} will now be listed
$math_axis\# + 3.51rth\# + {}^{36}/_{36}pt\# + subs * desc_depth\#$,	% num1
$math_axis\# + 1.51rth\# + {}^{9}/_{36}pt\#$,	% num2
$math_axis\# + 1.51rth\# + {}^{29}/_{36}pt\#$,	% num3
$-(math_axis\# - 3.51rth\# - subs * fig_height\# - {}^{74}/_{36}pt\#)$,	% denom1
$-(math_axis\# - 1.51rth\# - subs * fig_height\# - {}^{9}/_{36}pt\#)$,	% denom2
$6.49pt\# - subs * asc_height\#$,	% sup1
$5.99pt\# - subs * asc_height\#$,	% sup2
${}^{62}/_{36}pt\#$,	% sup3
${}^{36}/_{36}pt\#$, ${}^{72}/_{36}pt\#$,	% sub1, sub2
$subs * asc_height\# - {}^{36}/_{36}pt\#$, ${}^{18}/_{36}pt\#$,	% sup_drop, sub_drop
$11.9pt\#$, $8.1pt\#$,	% delim1, delim2

Finally, `cmsy5.mf` has these **fontdimen** parameters:

```
subs := 5/5;                                   % subscripts to 5pt will normally be in 5pt type
rth# := .4pt#;                                 % assume that rules will come from cmex10
fontdimen 8:                                   % parameters σ8 through σ22 will now be listed
  math_axis# + 3.51rth# + 36/36pt# + subs * desc_depth#,                    % num1
  math_axis# + 1.51rth# + 3/36pt#,                                          % num2
  math_axis# + 1.51rth# + 24/36pt#,                                         % num3
  − (math_axis# − 3.51rth# − subs * fig_height# − 63/36pt#),                % denom1
  − (math_axis# − 1.51rth# − subs * fig_height# − 3/36pt#),                 % denom2
  5.99pt# − subs * asc_height#,                                             % sup1
  5.49pt# − subs * asc_height#,                                             % sup2
  53/36pt#,                                                                 % sup3
  36/36pt#, 72/36pt#,                                                       % sub1, sub2
  subs * asc_height# − 36/36pt#, 18/36pt#,                                  % sup_drop, sub_drop
  9.9pt#, 7.1pt#,                                                           % delim1, delim2
```

There's also a bold symbols font, `cmbsy10`, which demonstrates one of many further possibilities. It has the 62 parameters of `cmmib10`, plus this:

```
subs := 7/10;                                  % subscripts to 10pt will normally be in 7pt type
fontdimen 8:                                   % parameters σ8 through σ22 will now be listed
  math_axis# + 3.51rule_thickness# + 54/36pt# + subs * desc_depth#,         % num1
  math_axis# + 1.51rule_thickness# + 30/36pt#,                              % num2
  math_axis# + 1.51rule_thickness# + 48/36pt#,                              % num3
  − (math_axis# − 3.51rule_thickness# − subs * fig_height# − 124/36pt#),    % denom1
  − (math_axis# − 1.51rule_thickness# − subs * fig_height# − 30/36pt#),     % denom2
  8.99pt# − subs * asc_height#,                                             % sup1
  8.49pt# − subs * asc_height#,                                             % sup2
  104/36pt#,                                                                % sup3
  54/36pt#, −(8.49pt# − 2subs * asc_height# − 3.1rule_thickness#),          % sub1, sub2
  subs * asc_height# − 36/36pt#, 18/36pt#,                                  % sup_drop, sub_drop
  23.9pt#, 10.1pt#,                                                         % delim1, delim2
```

We have now described 74 of the 75 standard font parameter files in the Computer Modern family. The last one is `cmex10.mf`, a so-called "math extension" font that provides the extralarge symbols needed in complicated formulas. It has exactly the parameters of `cmr10.mf`, except that its **font_identifier** is `"CMEX"`; furthermore, where `cmr10.mf` closes with '**generate** `roman`', the file `cmex10.mf` signs off as follows:

```
% the following font parameters are explained in The TEXbook, Appendix G
fontdimen 8:                                   % parameters ξ8 through ξ13 will now be listed
  rule_thickness#,                                                          % default_rule_thickness
  40/36pt#,                                                                 % big_op_spacing1
  60/36pt#,                                                                 % big_op_spacing2
  72/36pt#,                                                                 % big_op_spacing3
  216/36pt#,                                                                % big_op_spacing4
  36/36pt#;                                                                 % big_op_spacing5
generate mathex                                                             % switch to the driver file
```

Let's conclude this chapter by listing all 75 of the standard fonts whose parameters have been defined on the previous pages:

cmr17	cmbx8	cmss10	cmb10	cmbxti10
cmr12	cmbx7*	cmss9	cmff10	cmmib10
cmr10*	cmbx6	cmss8	cmfib8	cmitt10
cmr9	cmbx5*	cmssi17	cmti12	cmu10
cmr8	cmtt12	cmssi12	cmti10*	cmfi10
cmr7*	cmtt10*	cmssi10	cmti9	cmcsc10
cmr6	cmtt9	cmssi9	cmti8	cmtcsc10
cmr5*	cmtt8	cmssi8	cmti7	cmsy10*
cmsl12	cmsltt10	cmssdc10	cmmi12	cmsy9
cmsl10*	cmvtt10	cmssbx10	cmmi10*	cmsy8
cmsl9	cmtex10	cminch	cmmi9	cmsy7*
cmsl8	cmtex9	cmssq8	cmmi8	cmsy6
cmbx12	cmtex8	cmssqi8	cmmi7*	cmsy5*
cmbx10*	cmss17	cmdunh10	cmmi6	cmbsy10
cmbx9	cmss12	cmbxsl10	cmmi5*	cmex10*

The sixteen fonts marked with an asterisk (*) are used in the plain TEX macro package; in some sense they comprise a minimal working set for technical typing.

1. The Stem and other Fat Stroaks of Capital Romans
is five parts of forty and two (the whole Body:)
Or, (which is all one) one sixth part of the Heighth
of an Ascending Letter (as all Capitals are Ascendents) . . .

2. The Stem, and other Fat Stroaks of Capitals Italick,
is four parts of forty and two, (the Body.)

3. The Stem, and other Fat Stroaks of Lower-Case Roman,
is three and an half parts of forty and two, (the Body.)

— JOSEPH MOXON, *Mechanick Exercises* (1683)

Variation in width is more essential.
This can be increased till the o *becomes almost circular.*
The greater this roundness, the clearer the script,
and the greater the capacity for a
significant contrast between thin and thick strokes,
as of light and shade.

— GIAMBATTISTA BODONI, *Manuale Tipografico* (1818)

Driver Files

The command '**generate** `roman`' in a parameter file tells METAFONT to look at another file, `roman.mf`, for further instructions. The new file is called a "driver" because it takes charge of the subsequent activities; it makes sure that everything gets done decently and in order.

A driver file starts out by establishing pixel-oriented versions of the parameters. Then it reads in various program files, which actually create the characters of the font. (We will study the program files right after we finish looking at driver files.) Finally it contains a "kerning program" that improves the fit between certain pairs of letters.

There are eight standard driver files in the Computer Modern family, one for each major type of font layout. The most important is `roman.mf`, which is used by 44 of the 75 standard fonts. Therefore we shall look at the file `roman.mf` first; here is its complete text:

```
% The Computer Modern Roman family of fonts (by D. E. Knuth, 1979–1985)

if ligs > 1: font_coding_scheme := "TeX text";
  spanish_shriek = oct "074"; spanish_query = oct "076";
else: font_coding_scheme := if ligs = 0: "TeX typewriter text"
    else: "TeX text without f-ligatures" fi;
  spanish_shriek = oct "016"; spanish_query = oct "017"; fi

mode_setup; font_setup;

input romanu;                                   % upper case (majuscules)
input romanl;                                   % lower case (minuscules)
input greeku;                                   % upper case Greek letters
input romand;                                   % numerals
input romanp;                   % ampersand, question marks, currency sign
input romspl;                  % lowercase specials (dotless ı, ligature æ, etc.)
input romspu;                                   % uppercase specials (Æ, Œ, Ø)
input punct;            % punctuation symbols common to roman and italic text
input accent;                   % accents common to roman and italic text
if ligs > 1: input romlig; fi                   % letter ligatures
if ligs > 0: input comlig; fi             % ligatures common with italic text
if ligs ≤ 1: input romsub; fi                   % substitutes for ligatures

ligtable "!": "‘" =: spanish_shriek;
ligtable "?": "‘" =: spanish_query;
font_slant slant; font_x_height x_height#;
```

```
if monospace: font_normal_space 9u#;                    % no stretching or shrinking
  font_quad 18u#;
  font_extra_space 9u#;
else: font_normal_space 6u# + 2letter_fit#;
  font_normal_stretch 3u#; font_normal_shrink 2u#;
  font_quad 18u# + 4letter_fit#;
  font_extra_space 2u#;
  k# := −.5u#; kk# := −1.5u#; kkk# := −2u#;              % three degrees of kerning
  ligtable "k": if serifs: "v": "a" kern −u#, fi "w": "e" kern k#,
    "a" kern k#, "o" kern k#, "c" kern k#;
  ligtable "P": "A" kern kk#,
    "y": "o" kern k#, "e" kern k#, "a" kern k#, "." kern kk#, "," kern kk#;
  ligtable "F": "V": "W": if serifs: "o" kern kk#, "e" kern kk#, "u" kern kk#,
        "r" kern kk#, "a" kern kk#, "A" kern kkk#,
      else: "o" kern k#, "e" kern k#, "u" kern k#,
        "r" kern k#, "a" kern k#, "A" kern kk#, fi
    "K": "X": "O" kern k#, "C" kern k#,
      "G" kern k#, "Q" kern k#;
  ligtable "T": "y" kern if serifs: k# else: kk# fi,
    "Y": "e" kern kk#, "o" kern kk#,
      "r" kern kk#, "a" kern kk#, "A" kern kk#, "u" kern kk#;
  ligtable "O": "D": "X" kern k#,
      "W" kern k#, "A" kern k#,
      "V" kern k#, "Y" kern k#;
  if serifs: ligtable "h": "m": "n":
      "t" kern k#, "u" kern k#, "b" kern k#,
      "y" kern k#, "v" kern k#, "w" kern k#;
    ligtable "c": "h" kern k#, "k" kern k#; fi
  ligtable "o": "b": "p": "e" kern −k#, "o" kern −k#, "x" kern k#,
      "d" kern −k#, "c" kern −k#, "q" kern −k#,
    "a": if serifs: "v" kern k#, "j" kern u#, else: "r" kern k#, fi
    "t": "y" kern k#,
    "u": "w" kern k#;
  ligtable "A": if serifs: "R": fi "t" kern k#,
    "C" kern k#, "O" kern k#, "G" kern k#, "U" kern k#, "Q" kern k#,
    "L": "T" kern kk#, "Y" kern kk#, "V" kern kkk#, "W" kern kkk#;
  ligtable "g": "j" kern −k#;                                   % logjam
  ligtable "I": "I" kern −k#; fi                                % Richard III
                    % there are ligature/kern programs for "f" in the romlig file
                    % and for "-", "‘", and "’" in the comlig file
bye.
```

This driver file generates three different sets of 128 characters, depending on the setting of *ligs*. If *ligs* = 2, we get TEX's normal text font layout, and if *ligs* = 0 we get the normal typewriter font layout; both of these coding schemes are shown in full in the character indexes at the end of this book. If *ligs* = 1, the result is a compromise in which the characters '↑', '↓', '`', '<', and '>' appear instead of the f-ligatures.

The next driver file, `title.mf`, is simply a subset of `roman.mf`; it is used for just one of the standard fonts, namely `cminch`, whose characters have a *cap_height* equal to one inch. Inch-high letters take up a lot of space, and `cminch` is used only for special purposes. So `cminch` has been limited to 36 characters instead of the normal 128.

```
                                        % This makes a short font (caps and digits only)
font_coding_scheme := "ASCII caps and digits";
mode_setup; font_setup;
input romanu;                           % upper case (majuscules)
input romand;                           % numerals
font_slant slant; font_x_height x_height#;
if monospace: font_normal_space 9u#;    % no stretching or shrinking
  font_quad 18u#;
  font_extra_space 9u#;
else: font_normal_space 6u# + 2letter_fit#;
  font_normal_stretch 3u#; font_normal_shrink 2u#;
  font_quad 18u# + 4letter_fit#;
  font_extra_space 2u#;
  k# := -.5u#; kk# := -1.5u#; kkk# := -2u#;   % three degrees of kerning
  ligtable "P": "T": "Y": "A" kern kk#;
  ligtable "F": "V": "W": "A" kern if serifs: kkk# else: kk# fi,
    "K": "X": "O" kern k#, "C" kern k#,
      "G" kern k#, "Q" kern k#;
  ligtable "O": "D": "X" kern k#,
      "W" kern k#, "A" kern k#,
      "V" kern k#, "Y" kern k#;
  ligtable "A": if serifs: "R": fi
    "C" kern k#, "O" kern k#, "G" kern k#, "U" kern k#, "Q" kern k#,
    "L": "T" kern kk#, "Y" kern kkk#, "V" kern kk#, "W" kern kkk#;
  ligtable "I": "I" kern -k#; fi                % Richard III
bye.
```

Another simple variant of `roman.mf` is `texset.mf`, which produces 128 characters of an extended ASCII code:

```
% Special version of Computer Modern for TEX's extended ASCII character set
font_coding_scheme := "TeX extended ASCII";
mode_setup; font_setup; mono_charic# := 0;
input tsetsl;                           % special symbols to be slanted
slant := 0;                             % the remaining characters will not be slanted
currenttransform := identity yscaled aspect_ratio scaled granularity;
input romanu;                           % upper case (majuscules)
```

```
input romanl;                                   % lower case (minuscules)
input romand;                                   % numerals
input romanp;                   % ampersand, question marks, currency sign
input punct;            % punctuation symbols common to roman and italic text
input tset;                     % remaining special symbols of extended ASCII

font_x_height x_height#;
if monospace: font_normal_space 9u#;            % no stretching or shrinking
  font_quad 18u#;
  font_extra_space 9u#;
else: font_normal_space 6u# + 2letter_fit#;
  font_normal_stretch 3u#; font_normal_shrink 2u#;
  font_quad 18u# + 4letter_fit#;
  font_extra_space 2u#;
  k# := −.5u#; kk# := −1.5u#; kkk# := −2u#;     % three degrees of kerning
  ligtable "k": if serifs: "v": "a" kern −u#, fi "w": "e" kern k#,
    "a" kern k#, "o" kern k#, "c" kern k#;
  ligtable "P": "A" kern kk#,
    "y": "o" kern k#, "e" kern k#, "a" kern k#, "." kern kk#, "," kern kk#;
  ligtable "F": "V": "W": if serifs: "o" kern kk#, "e" kern kk#, "u" kern kk#,
          "r" kern kk#, "a" kern kk#, "A" kern kkk#,
       else: "o" kern k#, "e" kern k#, "u" kern k#,
          "r" kern k#, "a" kern k#, "A" kern kk#, fi
    "K": "X": "O" kern k#, "C" kern k#,
      "G" kern k#, "Q" kern k#;
  ligtable "T": "y" kern if serifs: k# else: kk# fi,
    "Y": "e" kern kk#, "o" kern kk#,
      "r" kern kk#, "a" kern kk#, "A" kern kk#, "u" kern kk#;
  ligtable "O": "D": "X" kern k#,
      "W" kern k#, "A" kern k#,
      "V" kern k#, "Y" kern k#;
  if serifs: ligtable "h": "m": "n":
      "t" kern k#, "u" kern k#, "b" kern k#,
      "y" kern k#, "v" kern k#, "w" kern k#;
    ligtable "c": "h" kern k#, "k" kern k#; fi
  ligtable "o": "b": "p": "e" kern −k#, "o" kern −k#, "x" kern k#,
      "d" kern −k#, "c" kern −k#, "q" kern −k#,
    "a": if serifs: "v" kern k#, "j" kern u#, else: "r" kern k#, fi
    "t": "y" kern k#,
    "u": "w" kern k#;
  ligtable "A": if serifs: "R": fi "t" kern k#,
    "C" kern k#, "O" kern k#, "G" kern k#, "U" kern k#, "Q" kern k#,
    "L": "T" kern kk#, "Y" kern kk#, "V" kern kkk#, "W" kern kkk#;
  ligtable "g": "j" kern −k#;                                   % logjam
  ligtable "I": "I" kern −k#;                                % Richard III
fi;

bye.
```

CAPS AND SMALL CAPS FONTS are generated by the driver file `csc.mf`, which is a more interesting variant of `roman.mf` because it has to set up the font parameters twice:

```
                                    % Caps and Small Caps in Computer Modern Roman

font_coding_scheme := if ligs = 0: "TeX typewriter text"
  else: "TeX text without f-ligatures" fi;
spanish_shriek = oct "016"; spanish_query = oct "017";

mode_setup; font_setup;
input romanu;                                        % upper case (majuscules)
input greeku;                                        % upper case greek letters
input romand;                                                        % numerals
input romanp;                          % ampersand, question marks, currency sign
input romspu;                                     % uppercase specials (Æ, Œ, Ø)
input punct;                % punctuation symbols common to all roman and italic
input accent;                          % accents common to roman and italic text
input romsub;                                        % substitutes for ligatures
if ligs > 0: input comlig; fi                      % ligatures common with italic

ligtable "!": "`" =: spanish_shriek;
ligtable "?": "`" =: spanish_query;
font_slant slant; font_x_height x_height#;
if monospace: font_normal_space 9u#;                % no stretching or shrinking
  font_quad 18u#;
  font_extra_space 9u#;
else: font_normal_space 6u# + 2letter_fit#;
  font_normal_stretch 3u#; font_normal_shrink 2u#;
  font_quad 18u# + 4letter_fit#;
  font_extra_space 2u#;
  k# := −.5u#; kk# := −1.5u#; kkk# := −2u#;          % three degrees of kerning
  ligtable "P": "T": "Y": "a" kern kk#, "A" kern kk#;
  ligtable "F": "V": "W": if serifs: "a" kern kkk#, "A" kern kkk#,
    else: "a" kern kk#, "A" kern kk#, fi
    "K": "X": "o" kern k#, "O" kern k#, "c" kern k#, "C" kern k#,
      "g" kern k#, "G" kern k#, "q" kern k#, "Q" kern k#;
  ligtable "O": "D": "x" kern k#, "X" kern k#,
      "w" kern k#, "W" kern k#, "a" kern k#, "A" kern k#,
      "v" kern k#, "V" kern k#, "y" kern k#, "Y" kern k#;
  ligtable "A": if serifs: "R": fi"c" kern k#, "C" kern k#,
      "o" kern k#, "O" kern k#, "g" kern k#, "G" kern k#,
      "u" kern k#, "U" kern k#, "q" kern k#, "Q" kern k#,
    "L": "T" kern kk#, "t" kern kk#, "Y" kern kk#, "y" kern kk#,
      "V" kern kkk#, "v" kern kkk#, "W" kern kkk#, "w" kern kkk#;
  ligtable "I": "I" kern −k#; fi                                   % Richard III
                    % ligature programs for "-", "`", and "'" are in the comlig file
```

```
forsuffixes $ = u, width_adj, cap_serif_fit, letter_fit,
  body_height, cap_height, x_height, bar_height, comma_depth,
  flare, cap_hair, stem, cap_stem, cap_curve, cap_ess,
  cap_jut, beak_jut, beak, slab, cap_bar, cap_band, cap_notch_cut,
  o, apex_o: $# := lower.$#; endfor
fudge := lower.fudge;
font_setup;                                   % now try again with lower settings
extra_endchar := extra_endchar & "charcode:=charcode+code_offset;";
code_offset := ASCII "a" − ASCII "A";
input romanu;                                 % majuscules (in lowercase positions)
code_offset := −3;
input romspu;                                 % Æ, Œ, Ø (in positions of æ, œ, ø)
code_offset := 0;
input cscspu;                  % substitutes for the remaining characters of romspl
if not monospace:
  k# := −.5u#; kk# := −1.5u#; kkk# := −2u#;          % three degrees of kerning
  ligtable "p": "t": "y": "a" kern kk#;
  ligtable "f": "v": "w": "a" kern if serifs: kkk# else: kk# fi,
    "k": "x": "o" kern k#, "c" kern k#,
      "g" kern k#, "q" kern k#;
  ligtable "o": "d": "x" kern k#,
      "w" kern k#, "a" kern k#,
      "v" kern k#, "y" kern k#;
  ligtable "a": if serifs: "r": fi"c" kern k#, "o" kern k#,
      "g" kern k#, "u" kern k#, "q" kern k#,
    "l": "'" kern kk#, "t" kern kk#, "y" kern kk#, "v" kern kkk#, "w" kern kkk#;
  ligtable "i": "i" kern −k#; fi                                   % SKIING
bye.
```

Computer Modern italic fonts come in two flavors, "text italic" and "math italic." The text italic driver, `textit.mf`, is very much like `roman.mf`:

```
                % The Computer Modern Text Italic family (by D. E. Knuth, 1979–1985)

mode_setup; font_setup;

input itall;                                  % lower case (minuscules)
input italsp;                  % lowercase specials (dotless ı, ligature æ, etc.)
if ligs > 1: font_coding_scheme := "TeX text";
  spanish_shriek = oct "074"; spanish_query = oct "076";
  input italig;                                                  % letter ligatures
else: font_coding_scheme := if ligs = 0: "TeX typewriter text"
    else: "TeX text without f-ligatures" fi;
  spanish_shriek = oct "016"; spanish_query = oct "017"; fi

font_slant slant; font_x_height x_height#;
if monospace: font_normal_space 9u#;                  % no stretching or shrinking
  font_quad 18u#;
  font_extra_space 9u#;
  letter_fit# := letter_fit := 0;
```

```
else: font_normal_space 6u# + 2letter_fit#;
  font_normal_stretch 3u#; font_normal_shrink 2u#;
  font_quad 18u# + 4letter_fit#;
  font_extra_space 2u#; fi

input romanu;                                          % upper case (majuscules)
input greeku;                                          % upper case greek letters
input itald;                                                              % numerals
input italp;                              % ampersand, question marks, currency sign
input romspu;                                   % uppercase specials (Æ, Œ, Ø)
input punct;                % punctuation symbols common to roman and italic text
input accent;                             % accents common to roman and italic text
if ligs > 0: input comlig; fi                  % ligatures common with roman text
if ligs ≤ 1: input romsub; fi                                % substitutes for ligatures

ligtable "!": "‘" =: spanish_shriek;
ligtable "?": "‘" =: spanish_query;

if not monospace:
  k# := −.5u#; kk# := −1.5u#; kkk# := −2u#;                % three degrees of kerning
  ligtable "d": "w": "l": "l" kern +u#;
  ligtable "F": "V": "o" kern kk#, "e" kern kk#,
      "u" kern kk#, "r" kern kk#, "a" kern kk#, "A" kern kkk#,
    "K": "X": "O" kern k#, "C" kern k#,
      "G" kern k#, "Q" kern k#;
  ligtable "T": "y" kern kk#,
    "Y": "e" kern kk#, "o" kern kk#, "r" kern kk#, "a" kern kk#, "u" kern kk#,
    "P": "W": "A" kern kk#;
  ligtable "O": "D": "X" kern k#, "W" kern k#, "A" kern k#,
      "V" kern k#, "Y" kern k#;
  ligtable "A": "R": "n" kern k#, "l" kern k#, "r" kern k#, "u" kern k#,
      "m" kern k#, "t" kern k#, "i" kern k#, "C" kern k#, "O" kern k#, "G" kern k#,
      "h" kern k#, "b" kern k#, "U" kern k#, "k" kern k#, "v" kern k#, "w" kern k#,
      "Q" kern k#, "L": "T" kern kk#, "Y" kern kk#, "V" kern kkk#, "W" kern kkk#,
    "b": "c": "e": "o": "p": "r": "e" kern −u#, "a" kern −u#, "o" kern −u#,
      "d" kern −u#, "c" kern −u#, "g" kern −u#, "q" kern −u#;
  ligtable "n": "’" kern kkk#; fi
                        % there are ligature/kern programs for "f" in the italig file
                                  % and for "-", "‘", and "’" in the comlig file
bye.
```

The math italic driver, `mathit.mf`, is quite different. In this case the kerning program includes provisions for positioning accents over the letters, assuming a `\skewchar` of *'177* (as explained in *The TEXbook*).

```
                % The Computer Modern Math Italic family (by D. E. Knuth, 1979–1985)

font_coding_scheme := "TeX math italic";
mode_setup; font_setup;

input romanu;                                          % upper case (majuscules)
```