The eurofont package v1.1.3

Rowland McDonnell
rebecca@astrid.u-net.com

30th January 1999

Contents

1 Introduction

The eurofont package was written to do two complementary jobs: firstly, to automate the process of using a euro symbol from any source in any fount[1]; secondly, to generate a 'faked' euro symbol from a C with two lines across it, which can be used automatically when no suitable real euro symbol exists.

Despite the name, the eurofont package does not itself have a fount containing euro symbols: it's meant to help you use founts with euro symbols.

The eurofont package defines two commands meant to be used in documents: the `\euro` command which prints a euro symbol, and the `\euros{`⟨*amount*⟩`}` command which prints a euro symbol next to its argument – normally a number – with a small space in between the two. The particular euro symbol printed depends on the way eurofont has been set up: the decision made is based on the fount in use at the time you use the command. The `\euros` command uses the `\euro` command to print the euro symbol, so these commands print the same symbol under the same circumstances.

There are several ways of controlling which particular euro symbol you get with any given fount; the idea is that – with a bit of luck – most users won't need to do anything more complicated than passing options to eurofont and making minor changes to the configuration file, `eurofont.cfg`.

Note for OzTeX users (and perhaps others): throughout this document, I refer to the standard set of PostScript Type 1 founts (Times, Helvetica, Courier, etc). If, like me, you use TrueType versions of Times, Palatino, etc., instead of PostScript Type 1 versions, the difference doesn't matter.

This is the second public release of the eurofont package; as I had expected, at least one bug did escape my notice in the first release (it was a problem with the advice given on configuring dvips). I've fixed the problem reported, but there's probably a few more bugs lurking in the package; if you do spot any bugs, have any trouble with the documentation, you'd like to tell me how to configure the dvi driver *you* use (I'd love to hear from Amiga or Archimedes TeX users in particular), or you'd just like to make a comment or suggestion about the eurofont package, I'd appreciate an email at: `rebecca@astrid.u-net.com`.

1.1 What else do I need?

The only thing you must have to use the eurofont package is a working LaTeX2e installation. This package has only been tested with the June 1998 release of

[1] I am one of the last people in the world to use this spelling of the word to refer to 'a set of type in one size and style'.

LaTeX2e, but will probably work correctly with earlier versions. Eurofont will not work with LaTeX 2.09.

The eurofont package can be configured to produce useful euro symbols in the absence of any founts containing real euro symbols, but you will need to edit the configuration file if you want this; an introduction to configuring this package can be found in section 1.5 on page 7.

In its default configuration, eurofont expects to find Adobe's Eurofonts installed. If you want to avoid these, you can pass the `marvosym` option to eurofont and it'll expect the Marvosym fount instead. Adobe's Eurofonts have italic, bold, and bold italic variants which Marvosym lacks.

Both Marvosym and the Eurofonts are PostScript Type 1 founts: you can use them if you have a PostScript printer, ATM (Adobe Type Manager), or a PostScript emulator like Ghostscript. Section 5.1 on page 20 has more on these founts and how to get them.

The eurosym and China2e packages both have euro symbols in Metafont format which can be used by eurofont. You'll get eurosym's euro symbol if you pass the `eurosym` option to eurofont – this gives you eurosym's euro symbol instead of Adobe's Eurofonts or Marvosym's euro symbols, and also replaces the euro symbols you get with the Computer Modern founts. You will need to edit the configuration file if you want to use the euro symbol from the China2e fount.

1.2 How to install the package – in brief

1. Run LaTeX on `eurofont.ins`.
2. Put `eurofont.sty`, `eurofont.cfg`, and all the files ending in `.fd` into a directory on your tex-inputs search path.
3. (optional) Get and install Adobe's Eurofonts, and/or the marvosym fount, and/or the China2e package, and/or the EC founts (the T1 encoded reworking of the original Computer Modern founts), and/or the Eurosym package. See section 5 on page 19 for details.
4. (optional) If you want to use Adobe's Eurofont's, put the `zpeu...tfm` files into a directory on your tex-fonts search path, and the `zpeu...vf` files into a directory on your vf (virtual founts) search path.
5. (optional) If you intend to use the marvosym fount, put one of the two files `fmvri8x.tfm` into a directory on your tex-fonts search path; put it with the file `fmvr8x.tfm` that comes with the marvosym distribution.
6. (optional) Copy the appropriate lines from `dvidrive.txt` to the appropriate file on your system to configure your dvi driver to use marvosym and/or Adobe's Eurofonts. This procedure is also covered in section 5.2 on page 21.
7. (optional) Modify the file `eurofont.cfg` as you like.

1.3 How to use the package – a brief introduction

This is how to use eurofont in your document:

```
\documentclass[a4paper]{article}
  \usepackage{eurofont}
```

```
\begin{document}
  The euro symbol looks like this: \euro.  A sum of money can
  be written like this: \euros{500}.
\end{document}
```

The `\euro` command prints the euro symbol; the `\euros` command is meant to be used to typeset a sum of money in euros: it prints the euro symbol to the left (by default) of its argument, with a small amount of space between the symbol and the text of the argument. If you give the eurofont package the `right` option, the euro symbol will be on the right.

The eurofont package's default setup assumes that you've got Adobe's Eurofonts installed. If you want use to euro symbols from the Marvosym fount instead, use the `marvosym` option:

```
\usepackage[marvosym]{eurofont}
```

The next question is: what do you get when you use the `\euro` or `\euros` command? The eurofont package decides which euro symbol to use depending on the fount in use at the place where the `\euro` command is encountered. The way the package is set up initially, you get this:

- All bar one of the 'standard' set of PostScript founts get a euro symbol from one of Adobe's Eurofonts: Bookman, Times, Palatino, New Century Schoolbook, and Utopia use Euroserif; Avant Garde, Helvetica, Symbol, and Zapf Dingbats use Eurosans; and Courier uses Euromono.
- Zapf Chancery (the one exception) uses a euro symbol faked with medium rules.
- Each of the three Computer Modern text fount families (Computer Modern Roman, Computer Modern Sanserif, and Computer Modern Typewriter (`cmr`, `cmss`, and `cmtt`) uses either the appropriate euro symbol from the matching 'text companion' (TC) fount (if installed); or, if the TC founts appear to be missing, eurofont will print a faked euro symbol instead.
- Everything else gets a euro symbol faked with medium weight rules.

The `marvosym` option tells eurofont to use the Marvosym fount's euro symbols in place of Adobe's Eurofonts.

The faked euro symbols I refer to above are euro symbols produced by superimposing a pair of horizontal lines over a letter 'C'. The result can be surprisingly tolerable in some cases.

1.3.1 Options

This is my best guess at the options most new users are likely to want to know about first, not a full list of all the options – you can find that in section 3 on page 11.

`left` This option makes the `\euros` command print the euro symbol to the left of its argument, which is normally a number; the `\euros` command is meant for typesetting sums of money. This is the default behaviour.

`right` This option makes the `\euros` command print the euro symbol to the right of its argument, which is normally a number; the `\euros` command is meant for typesetting sums of money.

`marvosym` This option tells eurofont to use the Marvosym fount's euro symbols for the standard set of PostScript Type 1 founts. The eurofont package automatically fakes a bold version when needed, and you'll also get a decent faked italic/slanted version if your dvi driver can slant a fount.

`eurosym` This option tells eurofont to use the eurosym package's euro symbol for the Computer Modern families and for the standard set of PostScript Type 1 founts. If the eurosym package isn't available, eurofont complains and all euro symbols are created using the normal eurofont code for printing faked euro symbols.

1.4 Some founts with euro symbols

It's a good idea to get some founts containing the euro symbol. Section 5 on page 19 has more details of some freely available founts. In brief, the founts with euro symbols in that I know about are:

Fount	**Format**	**Notes**
Adobe Eurofonts	PS Type 1	Seriffed, sanserif, and monospaced all in upright, italic, bold, and bold italic
Marvosym	pfa and pfb only	Seriffed, sanserif, and monospaced in upright medium only
China2e	Metafont	A single beautiful upright euro symbol
Eurosym	Metafont	The official euro and an alternative approach to printing a faked euro
Text Companion	Metafont	Founts containing extra symbols that come with the T1 encoded version of the Computer Modern founts

If you know of other founts containing euro symbols that are used with LaTeX do please let me know and I'll add them to this list.

The china2e and marvosym packages come with founts in one shape and weight only: medium upright. Despite this, the eurofont package will print what looks like a bold euro symbol from either fount, and can (with a bit of help from your dvi driver) also manage italic and bold italic euro symbols from marvosym.

The way it works is this: eurofont comes with two `tfm` files – both called `fmvri8x.tfm` – for a slanted version of the Marvosym fount[2]. This fount doesn't really exist, but some dvi drivers can create a slanted fount from an upright one by leaning it to one side.

Assuming your dvi driver can create a slanted fount (dvips can – see section 5.2.2 for more details), this extra `tfm` file means you can use the Marvosym fount's euro symbols with slanted or italic founts: the results you get with this should be good enough for all practical purposes. Eurofont also has two commands

[2]Use the `tfm` file from the `original` directory if you're using the original version of Marvosym (the `pfa` or `pfb` files from CTAN or the TrueType version from Martin Vogel's Web site); and use the `tfm` file from the `yandy` folder if you're using the Y&Y version of Marvosym – either downloaded from the Y&Y Web site, or the Macintosh version from CTAN.

to print a 'poor man's bold' version of a euro symbol; one of these is used automatically to print bold versions of the Marvosym or China2e euro symbols, which are missing bold variants. These poor man's bold commands print six copies of the requested symbol, each offset from the others by a very small amount in a hexagonal arrangement. There's a bit more detail in section 7.5.5 on page 42.

If you want to use the euro symbol from China2e, you could do worse than to read section 1.5.

1.5 How to change what you get – a brief intro

The eurofont package comes with a configuration file called `eurofont.cfg`. The idea is that you can change this file as much as you like to meet your preferences.

But before diving in and changing anything, there are several options that can change what the package does. There's a brief description of some of them in section 1.4 on the preceding page, and section 3 on page 11 tells all.

The basic idea behind eurofont's `\euro` command is this: there are several lists which have their contents defined in the file `eurofont.cfg`. When you use the `\euro` or `\euros` command, the current fount family is checked against each list in turn. If the fount family matches an entry in a list, then the command corresponding to that list is executed. This command prints a particular euro symbol. The first match you get decides which euro symbol you get: you can't get two euro symbols if one fount family is listed in, say, the `\chinaelist` and the `\seriflist`

So, for example, the fount family `ptm` (Adobe Times) is listed in `\seriflist`. If you use the `\euro` command in the middle of some text typeset in Times, the command `\makeserifeuro` is executed, and you get a euro symbol from Adobe's Euroserif fount (by default, that is; if you've used the `marvosym` option, you'll get Marvosym's seriffed euro symbol). I'll explain how to find out the internal fount family name of each fount in a bit.

The configuration file (`eurofont.cfg`) contains this by default:

```
%
%   List contents                                Corresponding command
%
   \EFaddtolist{\userlist}{}%                    \makeusereuro
   \EFaddtolist{\texteurolist}{}%                \maketexteuro
   \EFaddtolist{\chinaelist}{}%                  \makechinaeeuro
   \EFaddtolist{\cmlist}{cmr,cmss,cmtt}%         \makecmeuro
   \EFaddtolist{\seriflist}{pbk,pnc,ppl,ptm,put}% \makeserifeuro
   \EFaddtolist{\sanslist}{pag,phv,psy,pzd}%     \makesanseuro
   \EFaddtolist{\monolist}{pcr}%                 \makemonoeuro
   \EFaddtolist{\fakemediumlist}{pzc}%           \makefakemediumeuro
   \EFaddtolist{\fakelightlist}{}%               \makefakelighteuro
   \EFaddtolist{\fakeheavylist}{}%               \makefakeheavyeuro
```

The lists are created at the start of the eurofont package with nothing in them: the code above tells you the full story about what's in each list. You can use the `\EFaddtolist` command anywhere after the eurofont package has been loaded; you can use it in individual document preambles if you like, as well as in `eurofont.cfg`.

Each fount in LaTeX belongs to a fount family. For example, Bookman Roman, Bookman Italic, Bookman Bold, and Bookman Bold Italic are all different founts, but they belong to the Bookman family. This family has an internal LaTeX name:

pbk. One way of discovering the name of a fount family is to use eurofont's \showfontfamily command in your document: it displays the internal name of the current fount family on the screen and in the log file.

Before going any further, remember this: each list is a list of fount family names. If you're using a fount family that's in a particular list, you get the euro symbol generated by the command corresponding to that list. Section 6.1 on page 25 has more notes on LaTeX fount families.

If, for example, you don't like the euro symbol that you get with Computer Modern Roman, and you'd rather have the euro symbol from China2e with Computer Modern Roman, you can do this by changing two lines. Where the configuration file says this:

```
\EFaddtolist{\chinaelist}{}
\EFaddtolist{\cmlist}{cmr,cmss,cmtt}
```

you should change it to this:

```
\EFaddtolist{\chinaelist}{cmr}
\EFaddtolist{\cmrlist}{cmss,cmtt}
```

You might have a new PostScript fount that you'd like to use Adobe's Euroserif euro symbol with. If that fount is, say, Monotype Joanna (fount family name mjo), you can do this by adding mjo to the \seriflist:

```
\EFaddtolist{\seriflist}{pbk,pnc,ppl,ptm,put,mjo}
```

Or you might be using (say) Bitstream Bernhard Modern (family name bb7) which you might think doesn't go well with any of the real euro symbols on offer. You might prefer a euro symbol faked with light rules in this case. You can get this by saying:

```
\EFaddtolist{\fakelightlist}{bb7}
```

Then again, you might have a fount that has a real euro symbol in it: call it Adobe Xyzzy (pxy) for the sake of argument. If this fount has been set up properly for use with LaTeX, the euro symbol will be available using the \texteuro command. Rather than having to use a different command to select the euro symbol if you're using this fount family, you can say:

```
\EFaddtolist{\texteurolist}{pxy}
```

There are mechanisms that allow you to bypass the standard behaviours. You can, for example, use \newcommand in the configuration file to define any of the \make...euro commands to anything you like. You could, for example, say:

```
\newcommand{\makesanseuro}
    {{\fontfamily{phv}\selectfont\makefakeeuro}}
```

and all fount families in the \sanseurolist would have a euro symbol faked from a C in Helvetica with two rules drawn across it.

Alternatively, you could put a fount in \usereurolist, and you'd get a euro symbol generated by the \makeusereuro command. This is meant to be defined by you: by default, it prints a euro symbol faked with medium weight rules and displays on the screen a message explaining that you should have defined the

`\makeusereuro` command to do what you want it to do. For example, the following lines in the configuration file:

```
\newcommand{\makeusereuro}
    {EUR}
```

will give you 'EUR' – the standard international currency abbreviation for the euro currency unit – for the euro symbol in all fount families listed in the `\usereurolist`.

`\usereurolist` is the first list looked at: if a fount family is listed in the `\usereurolist` as well as another list, what you get is the euro symbol produced by `\makeusereuro`.

Immediately after `\usereurolist` is examined, the `\euro` command looks to see if a command of the form `\<fam>euro` exists (where `<fam>` is the name of the current fount family). If it does, this command is executed and the `\euro` command finishes. For example, assume you've designed a euro symbol to match URW's Arnold Boecklin fount (family name `uab`). Let's say you've written a command `\arnoldboecklineuro` to print this symbol. You could say this:

```
\newcommand{\uabeuro}{\arnoldboecklineuro}
```

and the `\euro` command would print your new euro symbol whenever you were using Arnold Boecklin.

1.6 About this package and document

The original aim of the eurofont package was to provide a trivial interface to allow one to use the euro symbols from Adobe's Eurofonts with any fount. The package has grown a bit since the original idea, which might explain a few things. While their contributions to the final code might look small, this package couldn't have been started (let alone finished) without the help of Donald Arseneau and Stefan Ulrich (in alphabetical order, in case you're wondering). They provided the list handling code which is at the heart of the package.

The biggest problem I had with this package was documenting it. I've no idea whether or not the documentation has 'hit the right note' – if you've any comments at all to make about the documentation (or the package for that matter), do please email me and let me know what you think. Comments, suggestions, and bug reports are all very much welcome. To say that writing the documentation was a headache is putting it mildly. If you read the documentation from start to finish, you'll notice that there's a certain amount of repetition. This is deliberate, and is meant to make it easier to use the documentation.

One problem that this package will face is that more and more founts containing euro symbols will appear as time goes on. With a bit of luck, it's flexible enough for you to be able to configure it to deal with these new founts. If not, or if you've had a thought about what might be done with one or more of these new founts, do please let me know by email – if it's practical, it would be nice to modify this package to make it more useful.

2 Installing the eurofont package

The eurofont package proper comes in two files: `eurofont.dtx` and `eurofont.ins`. Running LaTeX on `eurofont.dtx` produces this document, so I'll assume you've done that. Running LaTeX on `eurofont.ins` generates:

`dvidrive.txt`	Information on configuring dvi drivers to use Adobe's Eurofonts and the Marvosym fount.
`eurofont.cfg`	The configuration file: edit this file to change eurofont's behaviour.
`eurofont.sty`	The package file proper.
`uzmvs.fd`	LaTeX code for selecting the Marvosym fount.
`uzpeur.fd`	LaTeX code for selecting Adobe's Euroserif fount.
`uzpeuss.fd`	LaTeX code for selecting Adobe's Eurosans fount.
`uzpeutt.fd`	LaTeX code for selecting Adobe's Euromono fount.

`eurofont.cfg` and `eurofont.sty` should be put in a directory on your tex-inputs search path: they're both essential. If you're using either Adobe's Eurofonts or the Marvosym fount, read `dvidrive.txt`. Once you've used it, you can discard it.

The `fd` files are only needed if you're using the founts in question. What they're for is this: TeX can only typeset text in a particular fount if it has a `tfm` file corresponding to that fount to look at. LaTeX `fd` files contain code to select a `tfm` file when you ask for a particular fount using LaTeX commands. For example, you might ask for 'U encoded Adobe Euroserif in bold italic'. LaTeX would then look at the file `uzpeur.fd`, and discover that this request corresponded to `zpeubi.tfm`.

If you intend to use the Marvosym fount, put the file `uzmvs.fd` into a directory on your tex-inputs search path. This file is named so that it won't clash with the original Marvosym fount definitions: you don't have to worry about the marvosym package getting confused because the eurofont package uses the marvosym fount differently.

If you intend to use Adobe's Eurofonts, put the three `uzpeu...fd` files into a directory on your tex-inputs search path. If you already have files with these names from a different source, I suggest that you replace the older files with the `fd` files that come with eurofont. If this causes any problems, please let me know by emailing `rebecca@astrid.u-net.com`.

2.1 Using Marvosym and Adobe's Eurofonts

How to get these founts is covered in section 5 on page 19. To use them with the eurofont package, you need to install the appropriate `fd` files and configure your dvi driver, as explained above. Assuming you have installed the actual fount files on your computer, you then need `tfm` files for these founts: these are the files that tells TeX exactly what size each letter is, and allows it to produce output with a given fount.

The eurofont package comes with suitable `tfm` files for Adobe's Eurofonts, and two extra `tfm` files for the Marvosym fount. Which of these two `tfm` files you should use depends on which version of the Marvosym fount you have.

2.1.1 Using Adobe's Eurofonts

If you intend to use Adobe's Eurofonts, you should put all 12 files in the `adobeuro/tfmfiles/` directory into a tex-fonts directory. These files should exist happily alongside any other `tfm` files generated for Adobe's Eurofonts: being generated by afm2tfm, they should be identical to any others you might have. I suggest that you remove any other `tfm` files you might have for these founts unless you know that you need to keep them installed. If this advice causes you any problems, please email me and let me know.

2.1.2 Using the Marvosym fount

The Marvosym fount comes with a `tfm` file of its own, and the eurofont package includes two extra `tfm` files: you should use one of these to allow you access to a faked italic/slanted version of Marvosym, assuming that your dvi driver can produce a faked italic by slanting an upright fount. Both dvips and OzTeX can do this.

If you are using a version of the original Marvosym fount (the original `pfa` or `pfb` files from CTAN, or the Truetype version from Martin Vogel's Web site), then you should take the file `fmvri8x.tfm` from the directory `marvosym/tfmfiles/original`, and put it in the same tex-fonts directory as the file `fmvr8x.tfm` which came with the marvosym package.

If you are using one of the ATM-compatible versions of the Marvosym fount from Y&Y (either from Y&Y's Web site or any of the Mac versions from CTAN), things are slightly more complicated. In this case, you might be using the Y&Y supplied `tfm` file `marvosym.tfm` to use this fount. If so, leave `marvosym.tfm` alone: take the files `fmvr8x.tfm` and `fmvri8x.tfm` from the directory `marvosym/tfmfiles/yandy`, and put them in the tex-fonts directory which contains `marvosym.tfm`. If you have changed the name of the Y&Y supplied `tfm` file to `fmvr8x.tfm`, just add `fmvri8x.tfm`.

Note that you should not use the file `fmvr8x.tfm` supplied for the original Marvosym fount with any of the Y&Y versions of Marvosym, or vice versa: the founts are very, very similar, but differ in tiny details which affect the metrics file very slightly. You should therefore use the `tfm` file supplied with the particular fount you're using. If you're not sure of the source of the Marvosym `tfm` file already installed on your computer, you might prefer to replace it with the appropriate version of `fmvr8x.tfm` supplied with the eurofont package.

The original PostScript version of the Marvosym fount has the PostScript name `Martin_Vogels_Symbole`, while the Y&Y re-worked version has the PostScript name `Marvosym`; this leads to different dvi driver configuration file entries which should help you avoid any confusion over which version you've got.

3 All the options

Here are all of eurofont's options, presented in no particular order.

`left` This option makes the `\euros` command print the euro symbol to the left of its argument. When you use this option, it's also passed to the eurosym package, so that eurosym's `\EUR` command will also print the euro symbol to the left of its argument. This is the default behaviour.

`right` This option makes the `\euros` command print the euro symbol to the right of its argument. When you use this option, it's also passed to the eurosym package, so that eurosym's `\EUR` command will also print the euro symbol to the right of its argument.

`marvosym` This option tells eurofont to use the Marvosym fount's euro symbols for the standard set of PostScript Type 1 founts. The eurofont package automatically fakes a bold version when needed, and you'll also get a decent faked italic/slanted version if your dvi driver can slant a fount. This option counteracts the `adobeeurofonts` option, and can be over-ridden by the `eurosym` option.

`adobeeurofonts` This option tells eurofont to use Adobe's Eurofonts to supply the euro symbols for the standard set of PostScript Type 1 founts; this setting is used by default. This option counteracts the `marvosym` option, and can be over-ridden by the `eurosym` option.

`eurosym` This option tells eurofont to use the eurosym package's euro symbol for the Computer Modern families, and the standard set of PostScript Type 1 founts. If the eurosym package isn't available, eurofont complains and all euro symbols are created using the normal eurofont code for printing faked euro symbols.

`noeurosym` This option counteracts the `eurosym` option; if (for example) you have a configuration file that says `\ExecuteOptions{eurosym}`, you can tell eurofont *not* to use eurosym's euro symbols, and use the normal founts – typically Adobe's Eurofonts and the Text Companion founts. This is the default behaviour.

These next four options only affect what you get when you've given the eurofont package the `eurosym` option. See the eurosym package's documentation for more details.

`official` This option is passed to the eurosym package, and tells it to give you the official euro symbol.

`gen` This option is passed to the eurosym package, and tells it to give you a faked euro symbol.

`gennarrow` This option is passed to the eurosym package, and tells it to give you a faked euro symbol with narrow cross-strokes.

`genwide` This option is passed to the eurosym package, and tells it to give you a faked euro symbol with wide cross-strokes.

The following three options affect how eurofont produces faked euro symbols.

`noslantfakeeuro` Eurofont's faked euro symbols are produced with two rules of the same length.

`normalslantfakeeuro` Eurofont's faked euro symbols are produced with two rules of slightly different length: the lower rule is the shorter of the two. This is the default behaviour; it approximates the difference in length of the two rules in the official euro symbol.

`bigslantfakeeuro` Eurofont's faked euro symbols are produced with two rules of greatly different length: the lower rule is by far the shorter of the two. This was inspired by the China2e fount's euro symbol.

And finally, some options which don't seem to belong with anything else.

notextcomp This option tells the eurofont package not to load the textcomp package; it counteracts the textcomp option. You might want to use this option if you find that some characters or accents are unexpectedly wrong when using the eurofont package – this sort of thing can be caused by loading the textcomp package. See also the fixtieaccent option.

textcomp This option tells the eurofont package to load the textcomp package (part of the standard LaTeX distribution) if it is available. This package defines the \texteuro command (amongst other things). Trying to load textcomp is the default behaviour. One possibly unwanted effect of loading textcomp is that tie accents are typically messed up if you're using PostScript Type 1 founts; the fixtieaccent option can help out with this.

fixtieaccent This option counteracts the nofixtieaccent option: it makes eurofont define the tie accent to work the way it does as standard when the textcomp package hasn't been loaded. If you find tie accents no longer work when using eurofont, you can use either this option or the notextcomp option to fix things.

nofixtieaccent This option counteracts the fixtieaccent option: it stops eurofont defining the tie accent to work the way it does as standard when the textcomp package hasn't been loaded. This is the default behaviour.

debugreport This option tells the eurofont package to print all sorts of debugging information out when you use its commands. I added this option for my own benefit, but you might find it useful if you're configuring eurofont in strange fashions and not getting what you want.

nodebugreport This option tells the eurofont package not to print out any debugging information. This is the default behaviour.

For those who might be interested: the following lines are executed just before loading the configuration file:

```
\ExecuteOptions{adobeeurofonts}%       Use Adobe's Eurofonts
\ExecuteOptions{noeurosym}%            Don't use eurosym
\ExecuteOptions{left}%                 Euro symbol on left
\ExecuteOptions{normalslantfakeeuro}% Fake euros with slight slant
\ExecuteOptions{nodebugreport}%        No debugging reports
\ExecuteOptions{textcomp}%             Load the textcomp package
\ExecuteOptions{nofixtieaccent}%       Don't modify tie accents
```

You can therefore over-ride any of these defaults by placing a subsequent \ExecuteOptions statement in the configuration file. For example, to make the \euros command place the euro symbol on the right of the sum by default, add:

```
\ExecuteOptions{right}
```

to the configuration file, eurofont.cfg.

4 Configuring the eurofont package

There are three main mechanisms for changing the behaviour of eurofont's \euro command: passing an option to the eurofont package, adding a fount family name to one of eurofont's lists, and re-defining the `\make...` commands used by the `\euro` command to print euro symbols. Options have been covered in section 3 on page 11; the other two are covered here.

The eurofont package comes with a configuration file – `eurofont.cfg` – that's meant to be changed to match your preferences. If you want to add a fount family to a list, or re-define some commands, you can make the changes in the configuration file. You can also put code to do these jobs in a document file if you like. Please don't change `eurofont.sty` itself, unless you change its name to something else.

The way the `\euro` command works is this: when you use the `\euro` command, it compares the current fount family name with the contents of a series of lists. If the current fount family name is in a given list, the command corresponding to that list is executed. This prints a particular euro symbol, and the `\euro` command finishes. If you don't know what a fount family name is in LaTeX, have a look at section 6.1 on page 25.

The lists are created at the start of the eurofont package, and have fount family names entered into them by the `\EFaddtolist` command. There's more than just the lists I've mentioned in section 1.5, though. The configuration file contains the following lines by default:

```
%
%   List contents                                    Corresponding command
%
   \EFaddtolist{\userlist}{}%                        \makeusereuro
   \EFaddtolist{\texteurolist}{}%                    \maketexteuro
   \EFaddtolist{\chinaelist}{}%                      \makechinaeeuro
   \EFaddtolist{\cmlist}{cmr,cmss,cmtt}%             \makecmeuro
   \EFaddtolist{\seriflist}{pbk,pnc,ppl,ptm,put}% \makeserifeuro
   \EFaddtolist{\sanslist}{pag,phv,psy,pzd}%         \makesanseuro
   \EFaddtolist{\monolist}{pcr}%                     \makemonoeuro
   \EFaddtolist{\fakemediumlist}{pzc}%               \makefakemediumeuro
   \EFaddtolist{\fakelightlist}{}%                   \makefakelighteuro
   \EFaddtolist{\fakeheavylist}{}%                   \makefakeheavyeuro

   \EFaddtolist{\faketexteurolist}{}

% Put all the standard LaTeX weights (and likely extras)
% into one of the following lists:

   \EFaddtolist{\EFlightserieslist}{ul,el,l,ulc,elc,lc,ulx,elx,lx}
   \EFaddtolist{\EFmediumserieslist}
                         {m,mb,db,sb,mc,mbc,dbc,sbc,mx,mbx,dbx,sbx}
   \EFaddtolist{\EFboldserieslist}{b,bx,bc}
   \EFaddtolist{\EFultraboldserieslist}{eb,ub,ebc,ubc,ebx,ubx}
```

The point of the configuration file is that you should change it any way you like. But what do all these lines mean? In brief, what goes on is this: when you use the `\euro` command (and remember that the `\euro` command is used by the `\euros` command so this discussion applies to both commands), the current LaTeX fount family name is compared to the contents of each of the lists above, in the order given, with an extra test I'll mention in a bit. If, for example, you're using New Century Schoolbook (fount family name `pnc`), what happens with the default settings is this:

1. \userlist is examined. It's empty, so no match is found, and the command continues.

2. This is the extra test mentioned above: a test is made to see if the command \pnceuro exists (in general, the command \<fam>euro is looked for, where <fam> is the LaTeX fount family name of the current fount). If this command exists, it's executed and the \euro command terminates. In this case, the command doesn't exist, so the command continues.

3. \texteurolist is examined. It's empty, so the \euro command continues.

4. \chinaelist is examined. It's empty, so the \euro command continues.

5. \cmlist is examined. It's not empty: the current fount family is pnc, but the list contains cmr, cmss, and cmtt. No match is found, so the \euro command continues.

6. \seriflist is examined. It's not empty: the current fount family is pnc, and the list contains pbk, pnc, ppl, ptm, and put. A match is found, so \makeserifeuro – the command corresponding to this list – is executed, which prints a euro symbol from one of Adobe's Euroserif founts, and the \euro command finishes.

You might be wondering what the \faketexteuro list is for. If a fount family is listed in \faketexteuro, the \maketexteuro command will always execute the \makefakeeuro command to generate a faked euro symbol for that fount family. It prevents the \euro command from printing a euro symbol using the \texteuro command. I'm not sure that this is particularly useful, but it seemed like a good idea at the time.

Another unanswered question is: what happens if the \euro command has made all its tests and not printed a euro symbol? In this case, the \makedefaulteuro command is executed. By default, this command executes the \makefakeeuro command which prints a euro symbol faked with medium weight rules. You can of course re-define the \makedefaulteuro and \makefakeeuro commands any way you like.

The \EF...serieslist commands are used by the commands that print faked euro symbols; these commands are slightly more involved than you might think. I'll deal with this in detail in a bit.

4.1 The \make...euro commands

It might be that you want to do something that can't be done conveniently by just changing which fount families are listed in the lists above. If so, you'll probably find it most convenient to play around with the definitions of the various \make...euro commands; please do cut-and-paste the original definitions into the configuration file and modify them there if you think this will help. I used the \providecommand command to define all the \make...euro commands so that the definitions in eurofont.sty won't have any effect if the commands have been defined earlier. There's more detail on what these commands do in section 7 on page 27 and in the commented source code.

The \euro command works like this: the current fount family name is compared against the contents of a series of lists. If the current fount family is present in a particular list, the command corresponding to that list is executed. The first match ends the execution of the `\euro` command: you'll not get two euro symbols if the current fount family is in two lists. You can see what I mean in section 7.1 on page 29.

One anomaly is the second test made in the `\euro` command, directly after looking in `\userlist`: this second test doesn't check for a match in a list, but instead checks for the existence of a command `\<fam>euro`, where `<fam>` is the current fount family name. If this command exists, it's executed and the `\euro` command ends. For example, if you define:

```
\newcommand{\pcreuro}{EUR}
```

every time you use the `\euro` or `\euros` command while using Courier – which has the fount family name `pcr` – you'll get 'EUR' printed (the standard international abbreviation for the euro) rather than a euro symbol of any sort.

This table is a partial summary of what goes on:

List name	**Command executed**	**What you get by default**
`\userlist`	`\makeusereuro`	Fakes a euro and prints a warning.
	`\<fam>euro`	If it exists, `\<fam>euro` is executed.
`\texteurolist`	`\maketexteuro`	Prints a euro symbol from the current fount or a faked euro symbol.
`\chinaelist`	`\makechinaeeuro`	Prints the euro from the China2e fount.
`\cmlist`	`\makecmeuro`	Prints a euro symbol from the current fount or a faked euro symbol.
`\seriflist`	`\makeserifeuro`	Prints a euro from Adobe Euroserif.
`\sanslist`	`\makesanseuro`	Prints a euro from Adobe Eurosans.
`\monolist`	`\makemonoeuro`	Prints a euro from Adobe Euromono.
`\fakelightlist`	`\makefakelighteuro`	Fakes a euro with light weight rules.
`\fakemediumlist`	`\makefakemediumeuro`	Fakes a euro with medium weight rules.
`\fakeheavylist`	`\makefakeheavyeuro`	Fakes a euro with heavy weight rules.

What happens when you use the `\euro` command is that each list is considered in turn, starting with the `\userlist`. If the fount family being used at that point in the document is in that list, a euro symbol is generated by the specified command and that's the end of the command. If the fount family being used isn't in the first list, the next list is looked at and so on. If this explanation is less than crystal clear, you might find it useful to read section 7.1 on page 29.

If each list has been checked and the current fount family hasn't been found in any of them, the `\makedefaulteuro` command is executed. This produces a faked euro symbol: the eurofont package has commands to make these by drawing two rules over a letter C; they can be surprisingly acceptable particularly if made with a sanserif fount.

A brief description of what each of the `\make...euro` command does is:

`\makefakeeuro` This command doesn't correspond to a list, but is used by the `\makedefaulteuro` command and other parts of the eurofont package. What it does is this: it first checks to see if the command `\<fam>fakeeuro` exists, where `<fam>` is the current LaTeX fount family name. If this command exists, it's executed; if not, the current fount family name is checked against (in this order) the `\fakelightlist`, `\fakemediumlist`, and `\fakeheavylist`. If there's a match, it executes `\makefakelighteuro`, `\makefakemediumeuro`, or \makefakeheavyeuro (depending on which list had the match). If it finds no match, it executes \makefakemediumeuro.

The \makefake...euro commands produce euro symbols faked by placing a pair of rules over a letter ‘C’.

For example, if you were using Zapf Chancery, which has an internal LaTeX fount family name of pzc, and \makefakeeuro were executed, the first test would be to see if \pzcfakeeuro existed. Since eurofont does define this command, it exists, and is executed.

\makeusereuro Meant to be defined by you; by default, this command generates an on-screen warning and executes the \fakemediumruleeuro command which prints a euro symbol faked with medium rules.

\maketexteuro This command executes the \texteuro command if these three conditions are met: the current LaTeX fount family is not listed in the \faketexteurolist, the \texteuro command exists (it’s defined by the textcomp package which eurofont tries to load by default), and the fount definition file ts1<fam>.fd exists (where <fam> is the current LaTeX fount family name – this fd file will normally exist if the fount does have a euro symbol available). If these conditions are not met, it executes \makefakeeuro instead of \texteuro.

The \texteuro command, defined by textcomp, prints a euro symbol from the current fount, assuming everything’s working right; the \makefakeeuro command prints a euro symbol faked by placing two rules over a letter ‘C’.

\makechinaeeuro This command prints the euro symbol from the China2e fount. If the current fount series is listed as light or medium by being in either \EFlightserieslist or \EFmediumserieslist you’ll get the straight China2e euro symbol; if the current fount series is listed as bold or ultra bold by being in either \EFboldserieslist or \EFultraboldserieslist you’ll get a faked bold China2e euro symbol. If the current fount series isn’t listed in any of these lists, you’ll get a straight China2e euro symbol.

The \makechinaeeuro command checks for the china2e package: if it’s missing, you’ll get a warning message and a faked euro symbol.

\makecmeuro This command normally has the same effect as \maketexteuro: it executes the \texteuro command if the three conditions are met (see above, in the \maketexteuro entry on page 17); otherwise, it executes \makefakeeuro – this command fakes a euro symbol from a C over-printed with two rules. This behaviour can be changed using an option: if you’ve passed the eurosym option to eurofont, the \makecmeuro command prints Eurosym’s euro symbol instead.

\makeserifeuro This command normally prints a euro symbol from one of Adobe’s Euroserif founts (medium, italic, bold, or bold italic). If you’ve passed the marvosym option to eurofont, you’ll get a seriffed euro symbol from the Marvosym fount instead. Since Marvosym doesn’t have real bold euro symbols, you’ll get a faked bold euro symbol if the current fount series is listed in \EFboldserieslist or \EFultraboldserieslist (as with the \makechinaeeuro command).

If you've passed the `eurosym` option to `eurofont`, you'll get a euro symbol generated by `eurosym`'s `\euro` command: this option over-rides both the `marvosym` option and the default `adobeeurofonts` option.

The commands `\makeserifeuro`, `\makesanseuro`, and `\makemonoeuro` are very similar.

`\makesanseuro` This command normally prints a euro symbol from one of Adobe's Eurosans founts (medium, italic, bold, or bold italic). If you've passed the `marvosym` option to `eurofont`, you'll get a sanserif euro symbol from the Marvosym fount instead. Since Marvosym doesn't have real bold euro symbols, you'll get a faked bold euro symbol if the current fount series is listed in `\EFboldserieslist` or `\EFultraboldserieslist` (as with the `\makechinaeeuro` command).

If you've passed the `eurosym` option to `eurofont`, you'll get a euro symbol generated by `eurosym`'s `\euro` command: this option over-rides both the `marvosym` option and the default `adobeeurofonts` option.

The commands `\makeserifeuro`, `\makesanseuro`, and `\makemonoeuro` are very similar.

`\makemonoeuro` This command normally prints a euro symbol from one of Adobe's Euromono founts (medium, italic, bold, or bold italic). If you've passed the `marvosym` option to `eurofont`, you'll get a monospaced euro symbol from the Marvosym fount instead. Since Marvosym doesn't have real bold euro symbols, you'll get a faked bold euro symbol if the current fount series is listed in `\EFboldserieslist` or `\EFultraboldserieslist` (as with the `\makechinaeeuro` command).

If you've passed the `eurosym` option to `eurofont`, you'll get a euro symbol generated by `eurosym`'s `\euro` command: this option over-rides both the `marvosym` option and the default `adobeeurofonts` option.

The commands `\makeserifeuro`, `\makesanseuro`, and `\makemonoeuro` are very similar.

`\makefakelighteuro` First checks to see if the command `\<fam>fakeeuro` exists (where `<fam>` is the name of the current fount family); if so, it executes `\<fam>fakeeuro` (as with `\makefakeeuro`). If not, it executes `\fakelighteuro` – this gives you a euro symbol faked with light rules.

The particular weight (or thickness) of these rules varies depending on which `\EF...serieslist` the current fount series is in: the rules are lightest if the current fount series is listed in `\EFlightserieslist`, progressively heavier if listed in `\EFmediumseries` and `\EFboldseries`, and heaviest if listed in `\EFultraboldseries`. If the current fount series isn't listed in any of these lists, you get the rules you'd get if the current series were listed in `\EFmediumseries`.

This command is very similar to the commands `\makefakemediumeuro` and `\makefakeheavyeuro`.

`\makefakemediumeuro` First checks to see if the command `\<fam>fakeeuro` exists (where `<fam>` is the name of the current fount family); if so, it executes `\<fam>fakeeuro` (as with `\makefakeeuro`). If not, it executes

\fakemediumeuro – this gives you a euro symbol faked with medium weight rules.

The particular weight (or thickness) of these rules varies depending on which \EF...serieslist the current fount series is in: the rules are lightest if the current fount series is listed in \EFlightserieslist, progressively heavier if listed in \EFmediumseries and \EFboldseries, and heaviest if listed in \EFultraboldseries. If the current fount series isn't listed in any of these lists, you get the rules you'd get if the current series were listed in \EFmediumseries.

This command is very similar to the commands \makefakemediumeuro and \makefakeheavyeuro.

\makefakeheavyeuro First checks to see if the command \<fam>fakeeuro exists (where <fam> is the name of the current fount family); if so, it executes \<fam>fakeeuro (as with \makefakeeuro). If not, it executes \fakeheavyeuro – this gives you a euro symbol faked with heavy rules.

The particular weight (or thickness) of these rules varies depending on which \EF...serieslist the current fount series is in: the rules are lightest if the current fount series is listed in \EFlightserieslist, progressively heavier if listed in \EFmediumseries and \EFboldseries, and heaviest if listed in \EFultraboldseries. If the current fount series isn't listed in any of these lists, you get the rules you'd get if the current series were listed in \EFmediumseries.

This command is very similar to the commands \makefakemediumeuro and \makefakeheavyeuro.

\makedefaulteuro This command normally prints a faked euro symbol generated with the \makefakeeuro command.

If you've passed the eurosym option to eurofont, the \makedefaulteuro command prints a euro symbol generated by eurosym's \euro command. Otherwise, the \makedefaulteuro command prints a faked euro symbol generated with the \makefakeeuro command.

For the curious: eurofont defines the \ESeuro command to be whatever eurosym defined the \euro command to be. Eurofont can then define the \euro command to be something else, but eurosym's \euro code is still accessible by using the \ESeuro command.

5 Founts containing euro symbols

This section was written in October 1998: I know that there are founts other than the ones I've noted below that contain euro symbols, but these are the only ones I know of that are convenient to use with LaTeX. As far as I know, the founts currently included with 'euro compatible' versions of MS-Windows and the MacOS have a euro symbol that is close to the official euro symbol. This doesn't match the fount it's included with in most cases. It seems to me that there is nothing to be gained by using these euro symbols with LaTeX. I expect this situation will change in time, and matching euro symbols are designed for more founts.

Aside from the Text Companion founts which accompany the EC founts (the relatively new T1 encoded versions of the standard Computer Modern founts used by LaTeX– I think they are included with all recent LaTeX distributions), Metafont euro symbols are included in two other founts available from CTAN: China2e, which has a single very lovely euro symbol; and Eurosym, which has the official euro symbol. You can find out how to get all of these in section 5.3 on page 24.

The euro symbols in the Text Companion founts were designed before the final official euro symbol was decided on, and might be considered a bit eccentric by some.

I know of two sets of PostScript Type 1 founts containing euro symbols: Adobe's Eurofonts, a set of 12 founts providing seriffed, sanserif, and monospaced euro symbols in medium upright, italic, bold, and bold italic version; and the Marvosym fount, which has three euro symbols, very similar to the medium upright seriffed, sanserif, and monospaced euro symbols from Adobe.

Your TeX system's documentation should tell you if you can use PostScript Type 1 founts. Typically, you need either: a PostScript printer or PostScript interpreter on your computer such as Ghostscript; or Adobe Type Manager (ATM) installed on your computer and a dvi driver (such as OzTeX's) which can take advantage of this.

5.1 Getting Marvosym or Adobe's Eurofonts

Adobe's Eurofonts are available (October 1998) in a Mac version from here:

```
ftp://ftp.adobe.com/pub/adobe/type/mac/all/eurofont.sea.hqx
ftp://ftp-pac.adobe.com/pub/adobe/type/mac/all/eurofont.sea.hqx
```

Textures users on Macs should also download these files from CTAN:

```
systems/mac/textures/contrib/IdealFonts/EuroDefs.sit.hqx
systems/mac/textures/contrib/IdealFonts/README.IF
```

You should still download Adobe's Eurofonts separately.

Adobe's Eurofonts are available (October 1998) in a version suitable for MS-Windows PCs and Unix from here:

```
ftp://ftp.adobe.com/pub/adobe/type/win/all/eurofont.exe
ftp://ftp-pac.adobe.com/pub/adobe/type/win/all/eurofont.exe
```

These files are self-extracting archives on MS-Windows computers which can be decompressed on Unix computers with the `unzip` command.

The Marvosym fount is available from CTAN in `pfb` and `pfa` versions, and I gather these versions won't work with ATM. There are also Mac versions based on the Y&Y re-working of Marvosym mentioned below – a Truetype version and a Mac PS Type 1 version that will work with ATM.

The Marvosym fount and package are (October 1998) in this location at CTAN: site (October 1998):

```
/fonts/psfonts/marvosym/
```

A version which is apparently ATM compatible (implying the original version from CTAN isn't) is available from here:

```
http://www.YandY.com/download/marvosym.zip
```

Note that the metrics for this version of Marvosym are slightly different to the original; to use this version of Marvosym with `eurofont`, you should re-name the file `marvosym.tfm` to `fmvr8x.tfm` and put it on your tex-fonts path.

Martin Vogel's Web page contains a TrueType version of Marvosym, in a format suitable for MS-Windows computers:

`http://www.fh-bochum.de/fb1/vogel/marvosym.html`

5.2 Dvi driver configuration for Adobe's Eurofonts and the Marvosym fount

This section contains information on how to configure dvips and OzTeX to use Adobe's Eurofonts and the Marvosym fount. I believe that pdfTeX can use dvips entries.

You might find it useful to refer to the file `dvidrive.txt` that is created by running LaTeX on `eurofont.ins`: `dvidrive.txt` contains a plain text version of this dvi driver configuration information, so you can cut and paste the lines you need to the appropriate file on your computer.

If you have any problems with this information, or if you have information on how to configure other dvi drivers, please let me know by email.

5.2.1 Dvi driver configuration lines for Adobe's Eurofonts

This section contains information for configuring dvi drivers to use these PostScript Type 1 founts; what you need to do is add the given lines to a file so that your dvi driver knows which PostScript Type 1 fount file (and so on) corresponds to a particular `tfm` file in a `dvi` file. In the case of dvips, the file you add lines to is typically called `psfonts.map`. In the case of OzTeX, you will normally add lines to the `Default` configuration file – if you also use dvips, you'll add lines to `psfonts.map` as well.

Note that versions of dvips before v5.83 have trouble doing partial fount downloading with Adobe's Eurofonts. Because of this, I have listed the dvips psfont.map entries with '`<<`' and with '`<`': the '`<<`' entries prevent dvips attempting to do partial fount downloading with that particular fount.

The '`<<`' syntax does not work with all versions of dvips; it does work with dvips v5.78, but doesn't work with v5.70. If you're using a pre '`<<`' dvips, you should use the entries with '`<`' and don't use partial fount downloading with any document containing Adobe's Eurofonts; passing the `-j0` switch to dvips will prevent it from doing partial fount downloading if this is normally turned on.

I've asked the author of dvips when the '`<<`' syntax was first introduced, and he can't remember.

I am told that pdfTeX can do partial fount downloading using Adobe's Eurofonts.

Eurofont configuration lines for MS-Windows and Unix The following lines were supplied by Stefan Ulrich `<ulrich@cis.uni-muenchen.de>`, who tells me that you need to re-name the `PFB` files `pfb`.

If these lines work with computers other than those running MS-Windows and Unix, please let me know so I can change the documentation to suit.

The following entries are for dvips v5.78 and possibly some other versions:

```
zpeurs  EuroSans-Regular     <<_1______.pfb
zpeubs  EuroSans-Bold        <<_1B_____.pfb
zpeuris EuroSans-Italic      <<_1I_____.pfb
zpeubis EuroSans-BoldItalic  <<_1BI____.pfb
zpeurt  EuroMono-Regular     <<_2______.pfb
zpeubt  EuroMono-Bold        <<_2B_____.pfb
zpeurit EuroMono-Italic      <<_2I_____.pfb
zpeubit EuroMono-BoldItalic  <<_2BI____.pfb
zpeur   EuroSerif-Regular    <<_3______.pfb
zpeub   EuroSerif-Bold       <<_3B_____.pfb
zpeuri  EuroSerif-Italic     <<_3I_____.pfb
zpeubi  EuroSerif-BoldItalic <<_3BI____.pfb
```

The following entries are for dvips v5.70 or earlier, probably dvips v5.79 and above, and pdfTeX. Since that I don't use pdfTeX, this suggestion might not work – please let me know if you have any problems.

```
zpeurs  EuroSans-Regular     <_1______.pfb
zpeubs  EuroSans-Bold        <_1B_____.pfb
zpeuris EuroSans-Italic      <_1I_____.pfb
zpeubis EuroSans-BoldItalic  <_1BI____.pfb
zpeurt  EuroMono-Regular     <_2______.pfb
zpeubt  EuroMono-Bold        <_2B_____.pfb
zpeurit EuroMono-Italic      <_2I_____.pfb
zpeubit EuroMono-BoldItalic  <_2BI____.pfb
zpeur   EuroSerif-Regular    <_3______.pfb
zpeub   EuroSerif-Bold       <_3B_____.pfb
zpeuri  EuroSerif-Italic     <_3I_____.pfb
zpeubi  EuroSerif-BoldItalic <_3BI____.pfb
```

Eurofont configuration lines for Macs Macintosh `psfonts.map` entries for dvips v5.78 and possibly some other versions:

```
zpeur   EuroSerif-Regular     <<EuroSerReg
zpeuri  EuroSerif-Italic      <<EuroSerIta
zpeub   EuroSerif-Bold        <<EuroSerBol
zpeubi  EuroSerif-BoldItalic  <<EuroSerBolIta
zpeurs  EuroSans-Regular      <<EuroSanReg
zpeuris EuroSans-Italic       <<EuroSanIta
zpeubs  EuroSans-Bold         <<EuroSanBol
zpeubis EuroSans-BoldItalic   <<EuroSanBolIta
zpeurt  EuroMono-Regular      <<EuroMonReg
zpeurit EuroMono-Italic       <<EuroMonIta
zpeubt  EuroMono-Bold         <<EuroMonBol
zpeubit EuroMono-BoldItalic   <<EuroMonBolIta
```

Macintosh `psfonts.map` entries for dvips v5.70 or before, probably dvips v5.79 and above, and pdfTeX. Since that I don't use pdfTeX, this suggestion might not work – please let me know if you have any problems.

```
zpeur   EuroSerif-Regular     <EuroSerReg
zpeuri  EuroSerif-Italic      <EuroSerIta
zpeub   EuroSerif-Bold        <EuroSerBol
zpeubi  EuroSerif-BoldItalic  <EuroSerBolIta
zpeurs  EuroSans-Regular      <EuroSanReg
```

```
zpeuris EuroSans-Italic       <EuroSanIta
zpeubs  EuroSans-Bold         <EuroSanBol
zpeubis EuroSans-BoldItalic   <EuroSanBolIta
zpeurt  EuroMono-Regular      <EuroMonReg
zpeurit EuroMono-Italic       <EuroMonIta
zpeubt  EuroMono-Bold         <EuroMonBol
zpeubit EuroMono-BoldItalic   <EuroMonBolIta
```

OzTeX config file entries: these lines are normally added to the `Default` config file in OzTeX's `Configs` folder. These lines should work with any version of OzTeX from at least version 1.7 onwards. Please let me know if you have any problems.

```
zpeur    EuroSerif-Regular     "Euro Serif"      nil
zpeuri   EuroSerif-Italic      "Euro Serif"      nil i
zpeub    EuroSerif-Bold        "Euro Serif"      nil b
zpeubi   EuroSerif-BoldItalic  "Euro Serif"      nil bi
zpeurs   EuroSans-Regular      "Euro Sans"       nil
zpeuris  EuroSans-Italic       "Euro Sans"       nil i
zpeubs   EuroSans-Bold         "Euro Sans"       nil b
zpeubis  EuroSans-BoldItalic   "Euro Sans"       nil bi
zpeurt   EuroMono-Regular      "Euro Monospace"  nil
zpeurit  EuroMono-Italic       "Euro Monospace"  nil i
zpeubt   EuroMono-Bold         "Euro Monospace"  nil b
zpeubit  EuroMono-BoldItalic   "Euro Monospace"  nil bi
```

5.2.2 Dvi driver configuration lines for Marvosym

I have been greatly confused while trying to work out how to configure dvi drivers to use Marvosym. I think I've got it right, but I can't test on anything but my Mac. I'd appreciate an email if you find that any of the information below turns out to be wrong.

There are two versions of the Marvosym out there in network land: the original, and a version that's been re-worked by Y&Y. As far as I can work out, the Y&Y version of Marvosym has identical glyphs to the original, and differs in that it works properly with ATM and has better hinting. This means that the new version should render better at low and medium resolutions. In other words, I think the Y&Y version is better than the original. The metrics for the two versions are very slightly different to each other, so you should make sure you're using the `tfm` file that came with the version of the Marvosym fount that you're using.

Dvips `psfonts.map` entries for the original Marvosym fount:

```
fmvr8x  Martin_Vogels_Symbole                     <marvosym.pfb
fmvri8x Martin_Vogels_Symbole " .167 SlantFont" <marvosym.pfb
```

Dvips `psfonts.map` entries for the Y&Y 'ATM compatible' re-worked Marvosym fount:

```
fmvr8x   Marvosym                      <marvosym.pfb
fmvri8x  Marvosym " .167 SlantFont"  <marvosym.pfb
```

I have assumed that all dvi drivers that can use PostScript Type 1 founts can also produce fake italic founts by slanting an upright version. Because of this, I have made no provision for dvi drivers that can't do this. If you have a dvi driver that can't fake an italic, please let me know and I'll modify the eurofont package to take this into account, probably by adding another option and writing a new `fd` file.

Note for Macintosh users A Mac version of the Marvosym fount has been created in PostScript Type 1 and TrueType versions. It uses the Y&Y names as above, and these lines are be needed for dvips's `psfonts.map` file:

```
fmvr8x   Marvosym                     <Marvo
fmvri8x  Marvosym " .167 SlantFont"  <Marvo
```

And these lines for OzTEX's default config file:

```
fmvr8x    Marvosym     Marvosym     nil
fmvri8x   Marvosym     Marvosym     nil i
```

If a version of the original Marvosym fount is ever released for Macs, these lines will be needed for dvips's `psfonts.map` file:

```
fmvr8x  Martin_Vogels_Symbole                    <MartiVogSym
fmvri8x Martin_Vogels_Symbole " .167 SlantFont" <MartiVogSym
```

And these lines for OzTEX's default config file:

```
fmvr8x    MartiVogSym  "Martin Vogels Symbole"       nil
fmvri8x   MartiVogSym  "Martin Vogels Symbole"       nil i
```

5.3 Metafont founts containing euro symbols

The three Metafont sources of euro symbols that I know of are the Text Companion founts, the China2e fount, and the Eurosym fount. If you know of others, I'd appreciate an email to tell me about them.

The current (October 1998) version of LaTeX is set up to work with the relatively new T1 (and TS1) encoded versions of the usual Computer Modern founts. If you don't have the 'European Computer Modern' and 'Text Companion' founts – otherwise known as the EC and TC founts – it might be a good idea to get them now: the TS1 encoded Text Companion founts are the ones containing euro symbols.

If the above paragraph doesn't make much sense to you, you can find out if your LaTeX has access to the European Computer Modern and Text Companion founts by LaTeXing this file:

```
\documentclass{article}
  \usepackage[T1]{fontenc}
  \usepackage{textcomp}
  \begin{document}
    Hello world.  \texteuro.
  \end{document}
```

If you get no error messages, you have access to both the European Computer Modern (EC) and Text Companion (TC) founts.

If not, you can get these founts, as well as the Eurosym and China2e founts and packages, from these locations at your nearest CTAN site:

```
/fonts/ec/
/fonts/eurosym/
/macros/latex/contrib/supported/china2e/
```

6 Potentially useful extra information

Most of the ideas in the eurofont package are not my own invention, and are documented elsewhere. LaTeX's fount selection scheme, for example, is documented in the file `fntguide.tex` which is part of the standard LaTeX distribution. The file `simple-nfss.tex` kept in the `info/` directory at CTAN is also useful.

If you're interested in understanding the details of the source code, you'll probably find it useful to read `clsguide.tex` (also part of the standard LaTeX distribution), as well as Leslie Lamport's 'LaTeX, a document preparation system', 2nd edition, Addison-Wesley. Donald Knuth's 'The TeXbook', also published by Addison-Wesley, is probably essential reading. Goosens, Mittlebach, and Samarin's 'The LaTeX Companion', published by Addison-Wesley, is apparently a useful source of documentation on the internals of LaTeX; I've found the commented LaTeX source code an adequate substitute for this book.

6.1 Fount families and series

LaTeX 2_ε comes with a way of selecting founts called the New Fount Selection Scheme (NFSS). Each fount you use is specified by five things: encoding, family, series, shape, and size. A typical specification is `OT1/ptm/m/n at 12pt`. This means an `OT1` encoded version of `ptm` (Adobe Times) family, `m` (medium) series, `n` (normal upright roman) shape, in a size of 12pt. The point of the NFSS is to take a particular fount specification – like `OT1/ptm/m/n at 12pt` – and work out which `tfm` file should be used.

This job is essential, because as far as TeX is concerned, a `tfm` file *is* a fount: without a `tfm` file, you get no letters on the page. A `tfm` file is just a list of sizes of each letter and things like that; the actual letter shapes are kept elsewhere. The dvi driver needs to know which `tfm` file corresponds to which 'real' fount, which is what dvips's `psfonts.map` is for.

What the NFSS does is add a high-level interface to let you select a particular `tfm` file without having to deal with any of the awkward details – LaTeX has always had some sort of interface to allow this sort of thing, but the NFSS is much better than earlier efforts, and much, much easier to set up for a new set of founts.

The precise details of exactly which `tfm` file is to be used for each fount specification are contained in the various `fd` files installed on your computer (they're kept somewhere on the tex-inputs path). `ot1ptm.fd`, for example, is consulted in the above example. It says that `ptm7t.tfm` should be used for `OT1/ptm/m/n` at all sizes. This doesn't mean that you can only get Adobe Times in one size, because TeX will scale the fount to the appropriate size – the instruction to do this is included in the `fd` file.

You might be wondering what an encoding is. Well, when you type information into a computer, each character you type is given a number and this number is stored in memory: this number 'is' the character you have just typed. The relationship between the characters and numbers is called an encoding. ASCII encoding, for example, says that the letter 'A' has the number 65; 'B' has the number 66, and so on. In the case of LaTeX, you have two encodings: the input encoding – which is used to translate the input file into TeX's internal encoding (which you can forget about for now); and the output encoding, which is used to create the `dvi` file. For example, a Mac user might type an ö (number 194) into a `tex` file. Assuming that he'd said `\usepackage[applemac]{inputenc}`, that would result in the appropriate character appearing in the dvi file. This character would be represented in the dvi file by a number 246 if you were using a T1 encoded fount at that point. If you were using an OT1 encoded fount, it would be represented by a combination of character number 128 (umlaut) placed over character number 111 (the letter o). There are many other output and input encodings that are commonly used. It is, for example, common to use ASCII input encoding, so you have to type `\"{o}` to get ö. LaTeX usually sorts out all the details for you so you don't have to think about encodings that often.

From the point of view of configuring the eurofont package, you can probably ignore encodings. The two most common text fount encodings used with LaTeX are OT1 and T1: the original 7-bit standard encoding and the newer 8-bit standard encoding respectively.

What eurofont is most interested in is the fount family name. This comes after the encoding: OT1/ptm, and is usually a three letter abbreviation like ptm for Adobe Times. The full explanation of the naming scheme is presented in the fontname documentation[3]. In brief, the first letter of this family usually indicates the firm that made the fount. Adobe is indicated by a 'p' (for PostScript, because Adobe created PostScript). The next two letters indicates the name of the fount. Times is indicated by 'tm'. Not all fount families follow this pattern: the Computer Modern families pre-date the fontname naming scheme, so (for example) the Computer Modern Sanserif family has the name 'cmss'. And some fount families named according to the fontname scheme have very strange names. The Adobe Euroserif family has the name 'zpeur', the letter z being a prefix meaning 'bizarre'.

There's effectively an unlimited range of fount family names. One way of finding out what the name is of a particular fount family is using eurofont's `\showfontfamily` command in your document: it'll print the current fount family on the console and in the log file. For example, if you LaTeX this file:

```
\documentclass{article}
  \usepackage{otherfont}% A fictitious package to select a fount
  \usepackage{eurofont}
  \begin{document}
    Blah blah blah
   \showfontfamily
  \end{document}
```

You'll get a message in the log file and the terminal output telling you what the fount family used to typeset 'Blah blah blah' is.

Another way of finding out a particular fount family name is this: files for using different founts with LaTeX – such as times.sty – tend to contain lines like:

```
\renewcommand{\sfdefault}{phv}
\renewcommand{\rmdefault}{ptm}
\renewcommand{\ttdefault}{pcr}
```

This means that the phv (Adobe Helvetica) family is used for `\sffamily`, the ptm family (Adobe Times) family is used for `\rmfamily`, and the pcr family (Adobe Courier) is used for `\ttfamily`. This means you can find out which fount family is used by examining the package file you use to select it.

[3] Available from CTAN: /info/fontname/

If you're interested in the details of producing faked euro symbols, the fount series becomes relevant. The normally available fount series are just different weights:

ul	ultra light	sb	semi bold
el	extra light	b	bold
l	light	bx	bold extended
m	normal (medium)	eb	extra bold
mb	medium bold	ub	ultra bold
db	demi bold		

Of these, only two are selectable with normal commands. `\bfseries`, for example, selects the `bx` (bold extended) series; and `\mdseries` selects the `m` (normal 'medium' weight). Most fount families come in two different weights only. No normal founts are available in the range of weights just listed, although Multiple Master founts (those with a weight design axis) can be used in any number of different weights over almost any range. If you have produced a Multiple Master fount setup that gives you access to the full standard range of LaTeX fount series, you are clearly mad[4] and equally clearly don't need me to explain anything about the NFSS.

If you want to select something like ultrabold, one way is to say:

```
\fontseries{ub}\selectfont
```

A complication is that some fount families have condensed versions, and this has to be indicated by the fount series too. This means that the list of 'standard' series can be extended to include:

ulc	ultra light condensed	dbc	demi bold condensed
elc	extra light condensed	sbc	semi bold condensed
lc	light condensed	bc	bold condensed
mc	normal (medium) condensed	ebc	extra bold condensed
mbc	medium bold condensed	ubc	ultra bold condensed

Likewise, a fount family might have extended version of some weights (Computer Modern Roman's Bold Extended is an example of this), so the 'standard' list of series must also include variants like `ulx` for ultra light extended. This is only relevant to eurofont when it's asked to create a faked euro symbol: it has to know which series are bold (or whatever) to decide what thickness lines to draw across the C.

7 How eurofont works – in detail

This section is not meant to be a replacement for the rest of the documentation for TeXnical people. The idea is that this is an intermediate step between the usual user documentation and the rather eccentrically-commented source code – it's probably a good idea to read the earlier parts of this document before digging in to this section.

The eurofont package provides two commands meant to be used in documents for generating a euro symbol: `\euro` which just prints a euro symbol, and

[4] And probably called something like Melissa or Sebastian

`\euros{⟨amount⟩}` which prints a euro symbol (generated by the `\euro` command) next to the argument of the command, with a small space between the two. The euro symbol is to the left of the argument by default: if you pass the `right` option to the eurofont package, the euro symbol is printed to the right of the argument of the command.

The `\euros` command uses the `\euro` command to generate the euro symbol, so everything that applies to the `\euro` command also applies to the `\euros` command.

I see the package code as being in several main parts: the code for handling the lists, the `\euro` command and supporting `\make...` commands, the other commands for selecting and printing euro symbols, and the code to construct fake euro symbols. This is an entirely arbitrary division; I mention it in the hope that it might explain the way I've divided up this part of the documentation.

Note that the `\make...euro` commands are all meant to be called directly by the `\euro` command; the idea is to give you a 'top level' set of commands to modify in the configuration file any way you like, while still allowing access to any of the euro symbols the eurofont package is set up to work with by default: the `\make...euro` commands are all defined in terms of other commands, and it's these other commands that do the job of actually printing a euro symbol. You might, for example, change the `\makechinaeeuro` command to print 'EUR':

```
\newcommand{\makechinaeeuro}{EUR}
```

and use entries in the `\chinaeeuro` list to select this, rather than the euro symbol I intended. Even so, the 'lower level' command `\chinaeeuro` will be unaffected: it will still produce the euro symbol from the China2e fount, and you will still be able to use it. For example, you might say:

```
\newcommand{\pgyeuro}{\chinaeeuro}
```

to get a euro symbol from the China2e fount with Adobe Goudy (family name pgy)

A point that might be useful to know is this: the eurofont package doesn't define the `\euro` command when it's loaded using the `\usepackage{eurofont}` command. What happens is that eurofont defines the `\EFeuro` command to begin with: `\EFeuro` is just eurofont's `\euro` command under a different name. The eurofont package arranges for the `\euro` command that you use in your document to be created later. The way it's done is this: eurofont adds some code to the `\AtBeginDocument` hook. This code uses TeX's `\let` command to make the `\euro` command identical to `\EFeuro` at the `\begin{document}` command. Immediately before this, the code added by eurofont uses TeX's `\let` command again to make the `\oldeuro` command identical to the `\euro` command. This means that eurofont's definition of `\euro` will over-ride any earlier definitions, while saving any earlier definitions for you to use. This might also explain strange behaviour you encounter if trying to re-define `\euro` yourself in a document preamble. It's probably best to do something like this:

```
\documentclass{whateveryouwant}
  ...
  \usepackage{eurofont}
  \newcommand{\myeuro}{Some code}
  \AtBeginDocument{\let\euro\myeuro}
```

```
...
\begin{document}
```

The eurosym package is a special case: if you pass the eurosym option to eurofont, eurofont uses \let to make the \ESeuro command identical to eurosym's \euro command just after eurofont has loaded eurosym. This \ESeuro command appears in eurofont's code in various places: remember that it's defined to be whatever the eurosym package defined the \euro command to be.

7.1 The \euro command

The \euro command works like this: it looks at the current fount family (I shall call this <fam> for now – ptm for Adobe Times, cmr for Computer Modern Roman, and so on), and performs the following sequence of tests:

```
If <fam> is in \userlist, \makeusereuro
 else
 if \<fam>euro exists, \<fam>euro
  else
  if <fam> is in \texteurolist, \maketexteuro
   else
   if <fam> is in \chinaelist, \makechinaeeuro
    else
    if <fam> is in \cmlist, \makecmeuro
     else
     if <fam> is in \serifeurolist, \makeserifeuro
      else
      if <fam> is in \sanseurolist, \makesanseuro
       else
       if <fam> is in \monoeurolist, \makemonoeuro
        else
        if <fam> is in \makefakelighteurolist, \makefakelighteuro
         else
         if <fam> is in \makefakemediumeurolist, \makefakemediumeuro
          else
          if <fam> is in \makefakeheavyeurolist, \makefakeheavyeuro
           else \makedefaulteuro
           fi fi fi fi fi fi fi fi fi fi fi
```

In other words, it first looks to see if the current fount family is listed in the \usereurolist. If it is, it executes the \usereuro command and finishes. If not, it looks to see if a command \<fam>euro exists. For example, if you're typesetting with Adobe Times at that point (family name ptm), it'll look for the \ptmeuro command. If this command exists, it is executed and the \euro command finishes. If not, it looks to see if the current fount family is in the \texteurolist. If it is, the \texteuro command is executed, and the \euro command finishes. This continues to the final test: if the current fount family is in the \makefakeheavyeurolist, the \makefakeheavyeuro command is executed, and the \eurocommand finishes. If it's got to the end of the tests and no match has been found, the \defaulteuro command is executed.

It's up to you to define the \makeusereuro and \<fam>euro commands. All the other \make...euro commands may be re-defined as you see fit, but are defined by the eurofont package to produce euro symbols.

7.2 The \make...euro commands

What each of these commands does by default is this:

7.2.1 \maketexteuro

Executes either the \texteuro command or the \makefakeeuro command. The \texteuro command is normally defined by the textcomp package (part of the standard LaTeX distribution) to print a euro symbol from the TS1 encoded complement to the current fount. This TS1 encoded complement often does not exist, and even if it does exist, it usually (October 1998) does not have a euro symbol. The \makefakeeuro command is described in section 7.2.2. The eurofont package tries to load the textcomp package by default; see section 3 on page 11 for more details.

The decision on whether to execute \texteuro or \makefakeeuro is made like this: the \texteuro command is executed if these three conditions are met: 1) the current fount family is not listed in \faketexteurolist, 2) the textcomp package has been loaded, and 3) that the fount definition file ts1<fam>.fd exists (where <fam> is the current fount family name). That fd file is the thing that tells LaTeX where to find the symbol asked for in the \texteuro command defined by the textcomp package: if it doesn't exist, the current fount family certainly doesn't have a euro symbol available in the expected place.

The decision isn't foolproof: there's no way that TeX can check for the existence of a real glyph in a fount (the matter is complicated greatly by virtual founts and 'missing glyph' rule boxes), so \maketexteuro's checks mustn't be relied on.

7.2.2 \makefakeeuro

This command first checks to see if the command \<fam>fakeeuro exists, where <fam> is the name of the current fount family; if this command exists, it's executed. If not, \makefakeeuro then checks for the presence of the current fount family name in (in this order) \EFfakelightlist, \EFfakemediumlist, and \EFfakeheavylist. If it finds a match, it executes the corresponding \makefake...euro command; these are explained below. If it finds no match, it executes the \makefakemediumeuro command. \makefakeeuro doesn't appear directly in the definition of the \euro command.

7.2.3 \makechinaeeuro

Executes the \chinaeeuro command, which prints a fake bold euro symbol created from the China2e fount's euro symbol if the current \fontseries is listed in the \EFboldserieslist or the \EFultraboldserieslist, and a straight euro symbol from the China2e fount otherwise. The \chinaeeuro command is described in section 7.3.1 on page 32.

7.2.4 \makecmeuro

If you've given the eurosym option to the eurofont package and the eurosym package has been loaded successfully, \makecmeuro executes the \ESeuro command which prints the specified eurosym euro symbol, as described in section 7.2.4. Otherwise, the \cmeuro command is executed, which gives you either a euro symbol from the text companion founts, or a faked euro symbol; this faked euro symbol is made with light rules for Computer Modern Roman and medium weight rules for Computer Modern Sanserif and Typewriter. The `\cmeuro` command is described in section 7.3.2 on the following page.

7.2.5 `\makeserifeuro`

This executes the `\serifeuro` command, described in section 7.3.7 on page 34. By default, `\serifeuro` executes the `\zpeureuro` command, described in section 7.3.9 on page 34, which prints a euro symbol from Adobe's Euroserif fount. If you have given the `marvosym` option to eurofont and you have not used the `eurosym` option, the `\serifeuro` command will instead execute the `\marvosymserifeuro` command, described in section 7.3.14 on page 35: this will try to print a seriffed euro symbol from the Marvosym fount. If the current fount series is in `\EFboldserieslist` or `\EFultraboldserieslist`, you'll get a faked bold euro from the Marvosym fount. The default setup assumes that your dvi driver can print a faked italic version of the Marvosym fount.

If you've given the `eurosym` option to the eurofont package and the eurosym package has been loaded successfully, the `\serifeuro` command prints the specified eurosym euro symbol by executing the `\ESeuro` command. The `\ESeuro` command is described in section 7.3.4 on page 33.

7.2.6 `\makesanseuro`

This executes the `\sanseuro` command, described in section 7.3.6 on page 33. By default, `\sanseuro` executes the `\zpeusseuro` command, described in section 7.3.10 on page 34, which prints a euro symbol from Adobe's Eurosans fount. If you have given the `marvosym` option to eurofont and you have not used the `eurosym` option, the `\sanseuro` command will instead execute the `\marvosymsanseuro` command, described in section 7.3.13 on page 35: this will try to print a sanserif euro symbol from the Marvosym fount. If the current fount series is in `\EFboldserieslist` or the `\EFultraboldserieslist`, you'll get a faked bold euro from the Marvosym fount. The default setup assumes that your dvi driver can print a faked italic version of the Marvosym fount.

If you've given the `eurosym` option to the eurofont package and the eurosym package has been loaded successfully, the `\sanseuro` command prints the specified eurosym euro symbol by executing the `\ESeuro` command.

7.2.7 `\makemonoeuro`

This executes the `\monoeuro` command, described in section 7.3.8 on page 34. By default, `\monoeuro` executes the `\zpeutteuro` command, described in section 7.3.11 on page 34, which prints a euro symbol from Adobe's Euroserif fount. If you have given the `marvosym` option to eurofont and you have not used the `eurosym` option, the `\monoeuro` command will instead execute the `\marvosymmonoeuro` command, described in section 7.3.14 on page 35: this will try to print a monospaced euro symbol from the Marvosym fount. If the current fount series is in `\EFboldserieslist` or `\EFultraboldserieslist`, you'll get a faked bold euro from the Marvosym fount. The default setup assumes that your dvi driver can print a faked italic version of the Marvosym fount.

If you've given the eurosym option to the eurofont package and the eurosym package has been loaded successfully, the \sanseuro command prints the specified eurosym euro symbol by executing the \ESeuro command.

7.2.8 \makefakelighteuro

If the command \<fam>fakeeuro exists (where <fam> is the name of the current fount family), execute it. Otherwise, execute the \fakelighteuro command, which prints a euro symbol faked from a letter C with two light rules printed over it; you can find out more about this in section 7.4.2 on page 36. The \makefakelighteuro command is unaffected by the eurosym option.

7.2.9 \makefakemediumeuro

If the command \<fam>fakeeuro exists (where <fam> is the name of the current fount family), execute it. Otherwise, execute the \fakemediumeuro command, which prints a euro symbol faked from a letter C with two light rules printed over it; you can find out more about this in section 7.4.3 on page 37. The \makefakemediumeuro command is unaffected by the eurosym option.

7.2.10 \makefakeheavyeuro

If the command \<fam>fakeeuro exists (where <fam> is the name of the current fount family), execute it. Otherwise, execute the \fakeheavyeuro command, which prints a euro symbol faked from a letter C with two light rules printed over it; you can find out more about this in section 7.4.1 on page 36. The \makefakeheavyeuro command is unaffected by the eurosym option.

7.2.11 \makedefaulteuro

Executes the \makefakeeuro command. This usually prints a euro faked with medium weight rules. See section 7.2.2 on page 30 for more details.

7.3 Other commands to print euro symbols

7.3.1 \chinaeeuro

This prints the euro symbol from the China2e fount. There is only one euro character available, in the upright shape and medium weight only. This command uses the \SelectOnWeight command to print either a China2e euro symbol in its natural state (for light and medium weights), or a 'poor man's bold' version for bold and ultra bold weights.

The \SelectOnWeight command is described in section 7.5.2 on page 41.

7.3.2 \cmeuro

The \cmeuro command executes the \texteuro command to print a euro symbol from the TS1 encoded complement of the current fount if these three conditions are met: 1) the current fount family is not in the \faketexteurolist; 2) the command \texteuro exists; and 3) the file ts1<fam>.fd exists, where <fam> is the name of the current fount family. If all three conditions are not met, \cmeuro executes the \makefakeeuro command to print a faked euro symbol; this command is described in section 7.2.2 on page 30.

7.3.3 \EFeuro

This is the name under which the \euro command is defined originally; the eurofont package adds some code to the standard LaTeX \AtBeginDocument hook that makes the \euro command equivalent to the \EFeuro command (using the primitive TeX \let command). This code added to the \AtBeginDocument hook is executed, unsurprisingly, at the \begin{document} command. It's best not to use the \EFeuro command yourself unless you have a particular need to, but if you're writing a class or package file that uses eurofont, you might have such a need.

The \EFeuro command is identical to the \euro command, but they are different commands. If you want to change the \euro command in your document, you should do so after the \begin{document} command. One suggestion on how to do this is in section 7 on page 28. This way of re-defining the \euro command means you still have access to eurofont's original \euro command under the name \EFeuro. The \EFeuro command shouldn't be used directly in a LaTeX document. It's not that anything terrible will happen if you do, just that LaTeX convention states that commands with mixed-case names are meant for use by class and package writers only, and it seemed right to me to add this restriction to the \EFeuro command.

7.3.4 \ESeuro

If you tell the eurofont package to load the eurosym package, eurofont makes the \ESeuro command identical to the \euro command defined by eurosym. This means that eurosym's \euro command is still available as \ESeuro, even though eurofont re-defines the \euro command to be something completely different. It's probably best not to use the \ESeuro command unless you have no choice in the matter, as is often the case when writing package files.

In case you're wondering, the \ESeuro command will be the same as the \oldeuro command if eurofont has loaded eurosym.

7.3.5 \oldeuro

The eurofont package adds another bit of code to the \AtBeginDocument hook: this code not only sets the \euro command to be whatever \EFeuro has been defined as (as mentioned in section 7.3.3), but just before that, the code sets the \oldeuro command to be whatever the \euro command was at that instant. This means that any package which has defined a \euro command has not wasted its effort: its command is still available as \oldeuro.

As mentioned above, the \ESeuro command will be the same as the \oldeuro command if eurofont has loaded eurosym.

7.3.6 \sanseuro

Depending on what you've asked for in the options to the eurofont package, this command executes one of three commands. By default, or if you've used the adobeeurofonts option, it executes \zpeusseuro; if you've used the marvosym option, it executes `\marvosymsanseuro`; and if you've used the eurosym option (which over-rides the other two), it executes `\ESeuro`.

The `\zpeusseuro` command is described in section 7.3.10; the `\marvosymsanseuro` is described in section 7.3.13 on the following page; and the `\ESeuro` command is described in section 7.3.4 on the previous page.

7.3.7 `\serifeuro`

Depending on what you've asked for in the options to the eurofont package, this command executes one of three commands. By default, or if you've used the `adobeeurofonts` option, it executes `\zpeureuro`; if you've used the `marvosym` option, it executes `\marvosymserifeuro`; and if you've used the eurosym option (which over-rides the other two), it executes `\ESeuro`.

The `\zpeureuro` command is described in section 7.3.9; the `\marvosymserifeuro` is described in section 7.3.14 on the next page; and the `\ESeuro` command is described in section 7.3.4 on the preceding page.

7.3.8 `\monoeuro`

Depending on what you've asked for in the options to the eurofont package, this command executes one of three commands. By default, or if you've used the `adobeeurofonts` option, it executes `\zpeutteuro`; if you've used the `marvosym` option, it executes `\marvosymmonoeuro`; and if you've used the eurosym option (which over-rides the other two), it executes `\ESeuro`.

The `\zpeutteuro` command is described in section 7.3.9; the `\marvosymmonoeuro` is described in section 7.3.15 on the following page; and the `\ESeuro` command is described in section 7.3.4 on the previous page.

7.3.9 `\zpeureuro`

This command prints a euro symbol from one of Adobe's four Euroserif founts (Roman, Italic, Bold, or Bold Italic). The selection is done using LaTeX's usual fount selection mechanism.

7.3.10 `\zpeusseuro`

This command prints a euro symbol from one of Adobe's four Eurosans founts (Roman, Italic, Bold, or Bold Italic). The selection is done using LaTeX's usual fount selection mechanism.

7.3.11 `\zpeutteuro`

This command prints a euro symbol from one of Adobe's four Euromono founts (Roman, Italic, Bold, or Bold Italic). The selection is done using LaTeX's usual fount selection mechanism.

7.3.12 `\marvosymeuro{`⟨*char*⟩`}`

This command is used by `\marvosymserifeuro` (and friends) to print a euro symbol from the Marvosym fount. Like the `\chinaeeuro` command, it uses the `\SelectOnWeight` command – covered in section 7.5.2 on page 41 – to print a normal euro symbol for medium and light weight founts, and produces a 'poor man's bold' version when you're using a bold or ultra bold fount.

The `\marvosymeuro` command takes a single argument, which is the number of the character to be printed: 99 for the sanserif euro symbol, 100 for the monospaced euro symbol, and 101 for the seriffed euro symbol.

7.3.13 `\marvosymsanseuro`

The `\marvosymsanseuro` command prints a sanserif euro symbol from the Marvosym fount by executing `\marvosymeuro{99}`.

7.3.14 `\marvosymserifeuro`

The `\marvosymsanseuro` command prints a seriffed euro symbol from the Marvosym fount by executing `\marvosymeuro{101}`.

7.3.15 `\marvosymmonoeuro`

The `\marvosymsanseuro` command prints a monospaced euro symbol from the Marvosym fount by executing `\marvosymeuro{100}`.

7.3.16 `\cmrfakeeuro`

This command creates a euro symbol faked with light rules, using the `\fakelighteuro` command which is described in section 7.4.2 on the next page. Note that the `\euro` command will execute `\<fam>fakeeuro` (where `<fam>` is the current fount family) if it's asked to fake a euro symbol using any of the `\makefake...euro` commands. This means that you need to re-define `\cmrfakeeuro` if you want to change the sort of faked euro symbol you get with Computer Modern Roman; just putting `cmr` in (say) `\fakemediumlist` won't do the job.

7.3.17 `\cmssfakeeuro`

This command creates a euro symbol faked with medium rules, using the `\fakemediumeuro` command which is described in section 7.4.3 on page 37. Note that the `\euro` command will execute `\<fam>fakeeuro` (where `<fam>` is the current fount family) if it's asked to fake a euro symbol using any of the `\makefake...euro` commands. This means that you need to re-define `\cmssfakeeuro` if you want to change the sort of faked euro symbol you get with Computer Modern Sanserif; just putting `cmss` in (say) `\fakelightlist` won't do the job.

7.3.18 `\cmttfakeeuro`

This command creates a euro symbol faked with medium rules, using the `\fakemediumeuro` command which is described in section 7.4.3 on page 37. Note that the `\euro` command will execute `\<fam>fakeeuro` (where `<fam>` is the current fount family) if it's asked to fake a euro symbol using any of the `\makefake...euro` commands. This means that you need to re-define `\cmttfakeeuro` if you want to change the sort of faked euro symbol you get with Computer Modern Typewriter; just putting `cmtt` in (say) `\fakelightlist` won't do the job.

7.3.19 `\pzcfakeeuro`

This command creates a euro symbol faked with medium rules, using the `\fakemediumeuro[-0.1ex]` command. The optional argument lowers the height of the cross-strokes by 0.1 ex to match the C in Zapf Chancery.

Note that the `\euro` command will execute `\<fam>fakeeuro` (where `<fam>` is the current fount family) if it's asked to fake a euro symbol using any of the `\makefake...euro` commands. This means that you need to re-define `\pzcfakeeuro` if you want to change the sort of faked euro symbol you get with Zapf Chancery; just putting `pzc` in (say) `\fakelightlist` won't do the job.

7.4 Commands used to produce faked euro symbols

7.4.1 `\fakeheavyeuro[`⟨***lift***⟩`][`⟨***slant corr***⟩`]`

This is one of three similar commands: this command prints a euro symbol faked with heavy rules. It's executed by the `\euro` command if the current fount family is in the `\fakeheavylist`. It takes two optional arguments: the first one, `[`⟨*lift*⟩`]`, lifts the cross-strokes of the faked euro symbol by the specified length; the default length is 0 pt. The second optional argument, `[`⟨*slant corr*⟩`]`, is a percentage factor by which the nominal slant of the fount is multiplied by before calculating the horizontal position of the rules. Its default value is 100 (i.e., multiply the slant by $100/100 = 1$); this argument never has any effect on upright founts because they have no slant.

If you pass one optional argument to `\fakeheavyeuro`, it is interpreted as being the `[`⟨*lift*⟩`]` argument.

`\fakeheavyeuro` is defined to be one of these three:

Option	**Command**
`noslantfakeeuro`	`\heavyruleeuronoslant`
`normalslantfakeeuro`	`\heavyruleeuronorm`
`bigslantfakeeuro`	`\heavyruleeurobigslant`

The optional argument to `\fakeheavyeuro` is passed as the first argument (rule lift) to the given `\heavyruleeuro...` command; it's 0 ex by default. The second argument (slant correction factor) to the `\heavyruleeuro...` command is left blank, so it defaults to 100. See section 7.4.4 on the following page for more about this.

This command was written for the sake of symmetry more than anything else; I would be surprised if anyone found a real use for it.

7.4.2 `\fakelighteuro[`⟨***lift***⟩`][`⟨***slant corr***⟩`]`

This is one of three similar commands: this command prints a euro symbol faked with light rules. It's executed by the `\euro` command if the current fount family is in the `\fakelightlist`. It takes two optional arguments: the first one, `[`⟨*lift*⟩`]`, lifts the cross-strokes of the faked euro symbol by the specified length; the default length is 0 pt. The second optional argument, `[`⟨*slant corr*⟩`]`, is a percentage factor by which the nominal slant of the fount is multiplied by before calculating the horizontal position of the rules. Its default value is 100 (i.e., multiply the slant by $100/100 = 1$); this argument never has any effect on upright founts because they have no slant.

If you pass one optional argument to `\fakeheavyeuro`, it is interpreted as being the `[`⟨*lift*⟩`]` argument.

`\fakelighteuro` is defined by the options in `eurofont` to be one of these three; `\lightruleeuronorm` by default.

Option	**Command**
`noslantfakeeuro`	`\lightruleeuronoslant`
`normalslantfakeeuro`	`\lightruleeuronorm`
`bigslantfakeeuro`	`\lightruleeurobigslant`

The optional argument to `\fakelighteuro` is passed as the first argument (rule lift) to the given `\lightruleeuro...` command; it's 0 ex by default. The second argument (slant correction factor) to the `\lightruleeuro...` command is left blank, so it defaults to 100. See section 7.4.4 for more about this.

7.4.3 `\fakemediumeuro[`⟨***lift***⟩`][`⟨***slant corr***⟩`]`

This is one of three similar commands: this command prints a euro symbol faked with light rules. It's executed by the `\euro` command if the current fount family is in the `\fakemediumlist`. It takes two optional arguments: the first one, `[`⟨*lift*⟩`]`, lifts the cross-strokes of the faked euro symbol by the specified length; the default length is 0 pt. The second optional argument, `[`⟨*slant corr*⟩`]`, is a percentage factor by which the nominal slant of the fount is multiplied by before calculating the horizontal position of the rules. Its default value is 100 (i.e., multiply the slant by $100/100 = 1$); this argument never has any effect on upright founts because they have no slant.

If you pass one optional argument to `\fakeheavyeuro`, it is interpreted as being the `[`⟨*lift*⟩`]` argument.

`\fakemediumeuro` is defined by the options in `eurofont` to be one of these three; `\lightruleeuronorm` by default.

Option	**Command**
`noslantfakeeuro`	`\mediumruleeuronoslant`
`normalslantfakeeuro`	`\mediumruleeuronorm`
`bigslantfakeeuro`	`\mediumruleeurobigslant`

The optional argument to `\fakemediumeuro` is passed as the first argument (rule lift) to the given `\mediumruleeuro...` command; it's 0 ex by default. The second argument (slant correction factor) to the `\mediumruleeuro...` command is left blank, so it defaults to 100. See section 7.4.4 for more about this.

7.4.4 `\EFruleeuro`

The command generates a euro symbol from a 'C' with a pair of rules drawn on top of it. `\EFruleeuro` is not meant to be used in documents directly; if you need it, you should follow the `eurofont` package's example and use it in the definition of a new command to generate a euro symbol. The `\mediumruleeuronorm` command and others show you a way of doing this.

`\EFruleeuro` has six arguments:

`\EFruleeuro`
`{`⟨*backshift %age*⟩`}{`⟨*top rule width %age*⟩`}{`⟨*bottom rule width %age*⟩`}`

{⟨*rule thickness*⟩}{⟨*rule vertical spacing*⟩}{⟨*rule vertical offset*⟩}
{⟨*slant correction factor %age*⟩}

The first three arguments are numbers which are interpreted as a percentage of the width of a letter ‘C’ in the current fount. The next three arguments are normal LaTeX lengths. The final argument is another number that’s interpreted as a percentage: it’s the percentage by which the nominal slant of a fount (as specified in the tfm file) is multiplied by before working out the horizontal position of the cross-strokes of a faked euro symbol; if this slant correction factor argument is left blank, it defaults to 100 (multiply by $100/100 = 1$, so it has no effect), which is what it should be most of the time. This slant correction factor never has any effect on upright founts, since they have a slant of 0.

The {⟨*rule vertical offset*⟩} and {⟨*slant correction factor %age*⟩} arguments are needed because the letters as printed are sometimes not quite what the metrics files would have you believe.

The way the whole thing works is this: the \EFruleeuro command begins by putting a letter ‘C’ on the page. It then backs up by <backshift>, and draws the two rules. Each rule is <rule thickness> thick, and the centre lines of the rules are separated by a vertical distance of <rule spacing> (if this dimension is 0 pt, the rules are printed on top of each other and it appears that you only have one rule printed). The top rule has a horizontal length of <top rule width>, and the bottom rule has a horizontal length of <bottom rule width>. The rules are positioned half-way up the ‘C’, plus <rule vertical offset>; this parameter is needed because some letters have metrics that don’t quite tie up with reality. For example, the C in Zapf Chancery is actually lower than the metrics would suggest, so eurofont’s calculations go awry and a ‘hand-correction’ of -0.1 ex needs to be applied. The following code is executed by eurofont’s \pzcfakeeuro command:

```
\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{-0.1ex}{}
```

This creates a euro symbol faked from a ‘C’ with a pair of horizontal rules drawn across it.

These rules have their left-hand edge 110% of the width of the C to the left of the right-hand edge of the C. The top rule has a width of 80% of the width of the C; the bottom rule has a width of 72% of the width of the C.

The rules are 0.04 ex thick, and the distance between the centre lines of the rules is 0.27 ex. The centre line of the pair of rules is usually half way up the C according to the fount metrics; in this case, these rules are shifted down from this position by 0.1 ex.

If the C used by \EFruleeuro is an italic C, the code to place the rules takes this into account: it shifts the rules an appropriate amount to the right. The rules are also staggered slightly to match the slant of the C – rather than put both rules directly above each other at the specified position (taking the italic correction into account), the top rule is placed a touch to the right and the bottom rule a touch to the left.

And this brings me on to the final parameter: the slant correction factor. This is needed because some founts have italic or oblique versions in which the slant angle specified in the fount metrics file doesn’t match the real slant of the printed letters. I have no idea why this is the case, but I do know that it means that \EFruleeuro’s careful calculations to place the rules correctly produce ugly results when working with these slanted founts. This slant correction factor is the percentage by which the slant parameter is scaled by. If the rules are being printed too far to the left (so that they protrude out the back of the C too far), you might try something like this:

```
\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{0ex}{200}
```

which works very well with Adobe Optima – the line above tells the command to work on the basis that the slant is twice as great as specified in the metrics file. Since the fontinst-generated `tfm` file has the slant as 11° (1° less than in the `afm` file due to rounding errors), and the slant I measured is 21°, you can see that this should work well; if you have this fount youself, you will see that it does indeed work very well.

On the other hand, Bitstream Optima has a specified slant of 10°, but a measured slant of 5°. This means the rules are placed too far to the right, so they don't protrude out the back of the C far enough. The obvious thing to try in this case is:

```
\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{0ex}{50}
```

which is a definite improvement but far from perfect. This is because the italic C in this fount is positioned further to the left than expected for reasons which are rather involved. There's no easy and good way to deal with this without adding yet another parameter to the `\EFruleeuro` command, which I decided wasn't justified. In the case of Bitstream Optima, you might prefer the results you get with a slant correction factor of 25% or perhaps even 0%.

A final note: in an attempt to reduce the Byzantine complexity of the `\EFruleeuro` command to manageable levels, I have assumed that the cross-strokes do not extend beyond the right-hand limit of the letter 'C' which they're printed over. This shouldn't cause any problems unless you do something strange.

7.4.5 `\mediumruleeuronorm{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command uses the `\SelectOnWeight` command (see section 7.5.2 on page 41) to print a faked euro symbol intended to match a medium weight fount family like Times: thicker rules are used for bolder founts in the family, and thinner rules are used for lighter founts in the family. The two rules drawn across the C have slightly different lengths, approximately the same as in the official euro symbol.

`\mediumruleeuronorm` takes two arguments: the first, the length `{`⟨*lift*⟩`}`, raises the cross-stroke rules by the specified amount from their default position half way up the 'C'; and the second, the number `{`⟨*slant corr*⟩`}`, is interpreted as the percentage by which the specified slant of the current fount is multiplied before calculating the horizontal position of the cross-strokes. See the explanation of `\EFruleeuro` in section 7.4.4 on page 37 for a fuller description of these parameters with examples.

7.4.6 `\mediumruleeuronoslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command uses the `\SelectOnWeight` command (see section 7.5.2 on page 41) to print a faked euro symbol intended to match a medium weight fount family like Times: thicker rules are used for bolder founts in the family, and thinner rules are used for lighter founts in the family. The two rules drawn across the C have the same length.

`\mediumruleeuronoslant` takes two arguments: the first, the length `{`⟨*lift*⟩`}`, raises the cross-stroke rules by the specified amount from their default position half way up the 'C'; and the second, the number `{`⟨*slant corr*⟩`}`, is interpreted as the percentage by which the specified slant of the current fount is multiplied before calculating the horizontal position of the cross-strokes. See the explanation of `\EFruleeuro` in section 7.4.4 on page 37 for a fuller description of these parameters with examples.

7.4.7 `\mediumruleeurobigslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command uses the `\SelectOnWeight` command (see section 7.5.2 on the next page) to print a faked euro symbol intended to match a medium weight fount family like Times: heavier rules are used for bolder founts in the family, and lighter rules are used for lighter founts in the family. The two rules drawn across the C have quite different lengths, following the example of the China2e euro symbol.

`\mediumruleeurobigslant`takes two arguments: the first, the length `{`⟨*lift*⟩`}`, raises the cross-stroke rules by the specified amount from their default position half way up the 'C'; and the second, the number `{`⟨*slant corr*⟩`}`, is interpreted as the percentage by which the specified slant of the current fount is multiplied before calculating the horizontal position of the cross-strokes. See the explanation of `\EFruleeuro` in section 7.4.4 on page 37 for a fuller description of these parameters with examples.

7.4.8 `\lightruleeuronorm{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeuronorm` (see section 7.4.5 on the previous page), but uses lighter rules meant to match a fount family like Computer Modern Roman.

7.4.9 `\lightruleeuronoslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeuronoslant` (see section 7.4.6 on the preceding page), but uses lighter rules meant to match a fount family like Computer Modern Roman.

7.4.10 `\lightruleeurobigslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeurobigslant` (see section 7.4.7), but uses lighter rules meant to match a fount family like Computer Modern Roman.

7.4.11 `\heavyruleeuronorm{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeuronorm` (see section 7.4.5 on the preceding page), but uses heavier rules that aren't likely to match anything but a few display founts.

7.4.12 `\heavyruleeuronoslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeuronoslant` (see section 7.4.6 on page 39), but uses heavier rules that aren't likely to match anything but a few display founts.

7.4.13 `\heavyruleeurobigslant{`⟨*lift*⟩`}{`⟨*slant corr*⟩`}`

This command behaves the same as `\mediumruleeurobigslant` (see section 7.4.7 on the preceding page), but uses heavier rules that aren't likely to match anything but a few display founts.

7.5 Other supporting commands

7.5.1 `\showfontfamily`

This command displays the current fount family name on the console, and also notes the same information in the log file. It's meant to be used if you're configuring eurofont, but aren't sure what a particular fount family name is.

If you don't know what a fount family name is in LaTeX, have a look at section 6.1 on page 25.

7.5.2 `\SelectOnWeight {`⟨*light*⟩`}{`⟨*medium*⟩`}{`⟨*bold*⟩`}{`⟨*ultrabold*⟩`}`

This command takes four arguments: the first is executed if the current fount series is listed in `\lightserieslist`; the second if the current fount series is in the `\mediumserieslist`; the third if it's in the `\boldserieslist`; and the fourth if it's in the `\ultraboldserieslist`. If the current fount series isn't listed in any of these lists, the second argument (used for medium weight founts) is executed.

A typical use of `\SelectOnWeight` is this:

```
\newcommand{\mediumruleeuronorm}[2]{%
\SelectOnWeight%
{\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{72}{0.07ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{72}{0.14ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{72}{0.18ex}{0.27ex}{#1}{#2}}% ultra bold
}
```

This defines the `\mediumruleeuro` command so that it'll print one of four different faked euro symbols, depending on the weight of the current fount: lighter weight founts get a euro symbol faked with lighter rules; the heavier the fount, the heavier the rules. If the fount series isn't in any of the lists, you get the rules used for the medium weight series.

Section 4 on page 13 shows what's in the four different `\EF...serieslist` commands by default.

7.5.3 `\EFaddtolist{`⟨*list name*⟩`}{`⟨*items to add*⟩`}`

This command takes two arguments: the first argument is the name of a list (which must have already been defined as a TeX command), and the second argument is a comma delimited list of items to add to the list. For example:

```
\EFaddtolist{\sanslist}{pag,phv,psy,pzd}
```

adds four items to the list called `\sanslist`. The list ends up containing the given items with a special delimiter to make it easier to examine the list using TEX code. A check is made to ensure that each item is unique: if you tried to put (say) `ptm` (Adobe Times) onto the list twice, you'd only end up with one instance of `ptm` in the list. No warnings are made when this happens.

7.5.4 `\EFiftexteuroexists{`⟨*if true*⟩`}{`⟨*if false*⟩`}`

This command takes two arguments. It's used by `\maketexteuro` and `\cmeuro` like this:

```
\EFiftexteuroexists{\texteuro}{\makefakeeuro}}
```

What happens is that three tests are made: if the current fount family is not listed in the `\faketexteurolist`, the `\texteuro` command exists, and a TS1 encoded `fd` file exists for the current fount family, then the first argument is executed (`\texteuro` in this example). If any of these tests fails, then the second argument (`\makefakeeuro` in this example) is executed.

The idea is that if there's a good chance that the `\texteuro` command will print something useful, then the first argument is executed; the second argument is executed otherwise.

The `\texteuro` command is defined by the standard LATEX textcomp package which eurofont tries to load by default. The TS1 encoded `fd` file searched for is `ts1<fam>.fd`, where `<fam>` is the name of the current fount family.

7.5.5 `\EF@pmb{`⟨*text*⟩`}`

This command, not meant to be used in documents, is one of the two commands the eurofont package uses to print faked bold characters. What it does is print six copies of its argument, each copy offset from the others by a small amount in a hexagonal arrangement. The other command to do this job, `\EF@pmsb`, is identical except that it uses a slightly smaller vertical offset to produce better results with the Marvosym euro symbols.

This fakery is no substitute for a real bold symbol, but it's much better than the 'poor man's bold' used in the TeXbook and the standard LATEX bm package. The eurofont package's `\EF@pmb` command produces tolerable results over the full range I've tested it: sizes from 5 pt to 90 pt; whereas the alternative three copy poor man's bold doesn't work at all well at large sizes. Thanks are due to Donald Arseneau for suggesting a six copy method.

If you're interested, the actual code can be found in section 8.6 on page 54.

8 The code itself

To the interested reader: the code below grew rather than being designed and constructed according to a plan. This is reflected in the curious structure and eccentric comments; the comments were written for the chap who wrote the code, rather than anyone else. There's a fair bit of redundant code left (commented out) and quite a lot of debugging reports and development notes left in – these are me talking to myself, so don't be worried if you see something inexplicable.

One problem with the documented code is that the visual formatting I used means that quite a lot of lines are longer than 72 characters, so they won't fit within the normal width of the text. There are therefore rather a lot of overfull `\hboxes` below. I can't see any easy way round this, so I have cheated and set `\hfuzz` to a large value to suppress the warnings.

Say 'hello' to the nice LaTeX program. Who am I, and what do I need? This package might work with any version of LaTeX 2$_\varepsilon$, but I've only tested it with the June 1998 release. For best results, you need the ec founts andthe textcomp package (part of recent LaTeXs).

```
\NeedsTeXFormat{LaTeX2e}[1998/06/01]
\ProvidesPackage{eurofont}[1999/01/30 v1.1.3 A package for using euro
symbols]
```

Get all these defined before loading the config file which might want to use them.

```
\def\userlist{}
\def\texteurolist{}
\def\chinaelist{}
\def\cmlist{}
\def\seriflist{}
\def\sanslist{}
\def\monolist{}
\def\fakemediumlist{}
\def\fakelightlist{}
\def\fakeheavylist{}
\def\faketexteurolist{}
% begin with list of what's light, medium, bold, and ultra bold
\def\EFlightserieslist{}
\def\EFmediumserieslist{}
\def\EFboldserieslist{}
\def\EFultraboldserieslist{}
```

`\ifEF@debugreport` `\EF@debugrep`

Flag and command to allow option switching of debugging reports

```
\newif\ifEF@debugreport
\EF@debugreportfalse% Debugging reports off by default
\def\EF@debugrep#1{\ifEF@debugreport\typeout{eurofont: #1}\fi}
```

8.1 List handling code

Needed by the config file etc., which is why it's here.

```
\EFaddtolist{\listname}{comma delimited list of items to add}
\EF@checkiflisted{item}{\listname}
  this sets \ifEF@listed to true or false
```

Code from Stefan Ulrich <ulrich@cis.uni-muenchen.de>:

`\EF@checkiflisted`

```
\newif\ifEF@listed
%
\def\EF@checkiflisted#1#2{% check if element #1 is in list #2
\EF@listedfalse%
\edef\thiselem{#1}% changed to edef from def RJMM 1/9/98
\let\@@elt\@elt% Save \@elt (in case this command is executed
%                  somewhere strange)
```

```
\def\@elt##1{\def\testelem{##1}%
\ifx\thiselem\testelem\EF@listedtrue\fi}%
#2\let\@elt\@@elt}% execute list and restore \@elt
```

\EFaddtolist

```
\newcommand{\EFaddtolist}[2]{% #1 = list name;
%                            #2 = comma-delimited list of items to add
% \typeout{elements to add: #2}% debugging code; retain
%   \expandafter%%% this seems redundant
\EF@addtolist#2,:#1\end}
```

\EF@addtolist

```
\def\EF@addtolist#1,#2:#3\end{% #3 is the list now
\def\@tempcmda{#2}%
\ifx\@tempcmda\@empty%
   \EF@debugrep{Last elem: #1}\relax% if #2 is empty, do this.
    \EF@addMember{#1}{#3}%            if #2 is empty, do this
    \else
     \EF@debugrep{elem: #1}\relax% if #2 is not empty, do this
      \EF@addMember{#1}{#3}%       if #2 is not empty, do this
%     \expandafter%%% this seems redundant
      \EF@addtolist#2:#3\end\fi}%  if #2 is not empty, do this
```

\EF@addMember

```
\newcommand{\EF@addMember}[2]{%
    %%% add #1 only if it isn't yet in the list
    %%% it surely would be more efficient without the check...
    {\EF@checkiflisted{#1}{#2}\relax% to suppress space
    \ifEF@listed
    \EF@debugrep{#1 already in the list}\relax%
    \else
    \EF@debugrep{adding #1 to \string #2}\relax%
    \EF@rightappenditem{#1}{#2}\fi}}
```

\EF@rightappenditem

```
\newtoks\EF@tokb% token list register for temp use
\newcommand{\EF@rightappenditem}[2]%
{\@temptokena={\@elt{#1}}\EF@tokb=\expandafter{#2}%
\xdef#2{\the\EF@tokb\the\@temptokena}}% change the list globally
```

End Stefan's bit

8.2 Options

Some options: select between Adobe's Eurofonts (typographically better) or the Marvosym fount for `\serifeuro`, `\sanseuro`, and `\monoeuro`.

```
\newif\ifEF@marvosym\EF@marvosymfalse% Adobe Eurofonts by default
\newif\ifEF@eurosym\EF@eurosymfalse%   Don't use eurosym by default
\newif\ifEF@textcomp\EF@textcomptrue% Load the textcomp package if
%                                     possible (by default)
\newif\ifEF@fixtieaccent\EF@fixtieaccentfalse% Don't re-define tie
%                                     accent by default
%
```

```
\DeclareOption{marvosym}      {\EF@marvosymtrue}
\DeclareOption{adobeeurofonts}{\EF@marvosymfalse}
%
\DeclareOption{eurosym}  {\EF@eurosymtrue}
\DeclareOption{noeurosym}{\EF@eurosymfalse}
%
\DeclareOption{debugreport}{\EF@debugreporttrue}
\DeclareOption{nodebugreport}{\EF@debugreportfalse}
%
\DeclareOption{notextcomp}{\EF@textcompfalse}
\DeclareOption{textcomp}{\EF@textcomptrue}
%
\DeclareOption{fixtieaccent}{\EF@fixtieaccenttrue}
\DeclareOption{nofixtieaccent}{\EF@fixtieaccentfalse}
```

`\fakelighteuro` `\fakemediumeuro` `\fakeheavyeuro`

These commands were defined in the options: they're used by `\makefakemediumeuro`, `\makefakelighteuro`, and `\makefakeheavyeuro` commands later on.

They all take two optional arguments:

```
\fake...euro[lift][slant correction factor%]
```

The first argument is a normal dimension; the second is a percentage. Default values are 0 pt and 100 respectively. The lift is the amount by which the cross-strokes are raised above the nominal vertical centre line of the C; the slant correction factor is the factor by which the nominal slant of an italic or oblique C is multiplied by before working out where to put the cross-strokes. Some founts have metrics that disagree with reality quite wildly in terms of the specified italic angle. If the cross-strokes are too far to the right in the case of slanted versions of a fake euro, this second optional argument should be set to something less than 100 (50 or 25 are good values to try to begin with); if the cross-strokes are too far to the left, this second optional argument should be set to something more than 100 to begin with (200 is a good value to try to begin with).

```
\newcommand*{\fakelighteuro}[1][0ex]{\def\EF@tmprlift{#1}\@fakelighteuro}
\newcommand*{\fakemediumeuro}[1][0ex]{\def\EF@tmprlift{#1}\@fakemediumeuro}
\newcommand*{\fakeheavyeuro}[1][0ex]{\def\EF@tmprlift{#1}\@fakeheavyeuro}
```

`\@fakelighteuro` `\@fakemediumeuro` `\@fakeheavyeuro`

These commands are the ones that change depending on which sort of slant you want on the rules; it's done like this so you can effectively have two optional arguments for the `\fake...euro` commands, which is awkward to arrange using `\newcommand`. These commands are meant to be called only from the three `\fake...euro` commands above.

```
\DeclareOption{noslantfakeeuro} {%
    \def\EF@fakeslant{0}% in case anyone's interested
    \newcommand*{\@fakemediumeuro}[1][100]
                {\mediumruleeuronoslant{\EF@tmprlift}{#1}}
    \newcommand*{\@fakelighteuro}[1][100]
                {\lightruleeuronoslant{\EF@tmprlift}{#1}}
    \newcommand*{\@fakeheavyeuro}[1][100]
                {\heavyruleeuronoslant{\EF@tmprlift}{#1}}}
\DeclareOption{normalslantfakeeuro} {%
    \def\EF@fakeslant{1}% in case anyone's interested
    \newcommand*{\@fakemediumeuro}[1][100]
```

```
                {\mediumruleeuronorm{\EF@tmprlift}{#1}}
    \newcommand*{\@fakelighteuro}[1][100]
                {\lightruleeuronorm{\EF@tmprlift}{#1}}
    \newcommand*{\@fakeheavyeuro}[1][100]
                {\heavyruleeuronorm{\EF@tmprlift}{#1}}}
\DeclareOption{bigslantfakeeuro} {%
    \def\EF@fakeslant{2}% in case anyone's interested
    \newcommand*{\@fakemediumeuro}[1][100]
                {\mediumruleeurobigslant{\EF@tmprlift}{#1}}
    \newcommand*{\@fakelighteuro}[1][100]
                {\lightruleeurobigslant{\EF@tmprlift}{#1}}
    \newcommand*{\@fakeheavyeuro}[1][100]
                {\heavyruleeurobigslant{\EF@tmprlift}{#1}}}
```

The `\euros` command was more-or-less stolen from eurosym.sty: The Euro Symbol Package for LaTeX by Henrik Theiling `<theiling@coli.uni-sb.de>`,

```
http://www.coli.uni-sb.de/~theiling
\ProvidesPackage{eurosymbol}
    [1998/08/06 v1.1 European currency symbol ``Euro'']
```

```
\DeclareOption{left}{\PassOptionsToPackage{left}{eurosym}%
\DeclareRobustCommand{\euros}[1]{\euro\nobreak\,#1}}
\DeclareOption{right}{\PassOptionsToPackage{right}{eurosym}%
\DeclareRobustCommand{\euros}[1]{#1\nobreak\,\euro}}
\DeclareOption{official}{\PassOptionsToPackage{official}{eurosym}}
\DeclareOption{gen}{\PassOptionsToPackage{gen}{eurosym}}
\DeclareOption{gennarrow}{\PassOptionsToPackage{gennarrow}{eurosym}}
\DeclareOption{genwide}{\PassOptionsToPackage{genwide}{eurosym}}
%
\ExecuteOptions{adobeeurofonts}% Use Adobe's Eurofonts by default
\ExecuteOptions{noeurosym}%      Don't use eurosym by default
\ExecuteOptions{left}%           Euro symbol on left by default
\ExecuteOptions{normalslantfakeeuro}% Fake euros with slight slant by default
\ExecuteOptions{nodebugreport}%  No debugging reports by default
\ExecuteOptions{textcomp}%  Load the textcomp package by default
\ExecuteOptions{nofixtieaccent}%  Don't re-define the tie accent by
%                                 default
```

```
\InputIfFileExists{eurofont.cfg}{}%
{\PackageWarningNoLine{eurofont}%
{I can't find the eurofont.cfg configuration file.\MessageBreak
Perhaps something is wrong with this installation?\MessageBreak
The \protect\euro\space command will work with default settings}%
\EFaddtolist{\userlist}{}
\EFaddtolist{\texteurolist}{}
\EFaddtolist{\chinaelist}{}
\EFaddtolist{\cmlist}{cmr,cmss,cmtt}
\EFaddtolist{\seriflist}{pbk,pnc,ppl,ptm,put}
\EFaddtolist{\sanslist}{pag,phv,psy,pzd}
\EFaddtolist{\monolist}{pcr}
\EFaddtolist{\fakemediumlist}{pzc}
\EFaddtolist{\fakelightlist}{}
\EFaddtolist{\fakeheavylist}{}
%
\EFaddtolist{\faketexteurolist}{}
```

The following lists are used to decide which thickness line should be used for producing faked euro symbols. demibold and semibold are in the medium list because I reckon it's better to have a line that's a touch too light than too heavy.

```
\EFaddtolist{\EFlightserieslist}{ul,el,l,ulc,elc,lc,ulx,elx,lx}
\EFaddtolist{\EFmediumserieslist}{m,mb,db,sb,mc,mbc,dbc,sbc,mx,mbx,dbx,sbx}
\EFaddtolist{\EFboldserieslist}{b,bx,bc}
\EFaddtolist{\EFultraboldserieslist}{eb,ub,ebc,ubc,ebx,ubx}%
%
% And two non-standard series:
%
\EFaddtolist{\EFultraboldserieslist}{xb,ebd}
}
```

Note: the stuff above should include more-or-less everything in the default config file.

```
\ProcessOptions
```

If requested (as it is by default), load textcomp.sty if it exists; this conditional code and subsequent checking for `textcomp.sty` should allow eurofont.sty to work with old or OT1-only LaTeXs.

```
\ifEF@textcomp\IfFileExists{textcomp.sty}{\RequirePackage{textcomp}}{}\fi
```

If the textcomp package is going to be loaded, it'll be loaded by now, so now is the time to re-define the tie accent (if this has been asked for):

```
\ifEF@fixtieaccent \DeclareTextAccentDefault{\t}{OML}\fi
```

```
\ifEF@eurosym%
```

even if eurosym can't be loaded, ensure that some sort of euro is printed:

```
\def\euro{\makefakeeuro}
```

Load eurosym if asked and if possible

```
\IfFileExists{eurosym.sty}{\RequirePackage{eurosym}}{%
\PackageError{eurofont}{I can't find the eurosym package}%
{You've used the eurosym option; this requires the eurosym package
which doesn't appear to be installed}}%
```

Save eurosym's `\euro` as `\ESeuro`; if eurosym couldn't be loaded, `\euro` is still defined as `\makefakeeuro` (see above), so a sort of `\euro` symbol will be printed

```
\let\ESeuro\euro
\fi
```

8.3 A spare command or more?

`\showfontfamily`

```
\newcommand{\showfontfamily}{
\typeout{**********************************}
\typeout{* \protect\showfontfamily:
      \space\space\space\space\space\space
      \space\space\space\space\space\space
      \space\space\space*}
\typeout{*\space\space\space\space\space\space
           \space\space\space\space\space\space
      \space\space\space\space\space\space
      \space\space\space\space\space\space
```

```
      \space\space\space\space\space\space
      \space\space\space*}
\typeout{* The current font family is: \f@family\space *}
\typeout{*\space\space\space\space\space\space
          \space\space\space\space\space\space
      \space\space\space\space\space\space
      \space\space\space\space\space\space
      \space\space\space\space\space\space
      \space\space\space*}
\typeout{***********************************}
}
```

8.4 Code to fudge a euro if needed

The author commands to generate faked euro symbols now select one of four weights of cross-stroke to match light, medium, bold, and ultrabold. This should do.

\EF@crossstrokes Arguments as noted; this is the command that draws the horizontal strokes for a fake euro symbol.

```
\def\EF@crossstrokes#1#2#3#4#5#6#7#8#9{%
%                   #1 = width of C
%                   #2 = height of C
%                   #3 = depth of C
%                   #4 = back shift as percentage of #1
%                   #5 = strokelen as percentage of #1
%                   #6 = strokelen of bottom rule as percentage of #1
%                   #7 = thickness of cross-strokes
%                   #8 = centre line spacing of cross-strokes
%                   #9 = extra rule lift
% Need to lift cross strokes by 1/2 (height C - height cross-strokes)
%
% N.B. slanted founts are an interesting thingy.
% The metrics often lie about the slant, so \EF@slantcorr is a
% percentage to multiply the slant/point figure from the tfm file by.
% Start out by ensuring that it exists:
\@ifundefined{EF@slantcorr}{\def\EF@slantcorr{100}}{}%
% \fontdimen1 is slant per point (in points).
\@tempdima=#8%
% \typeout{rule spacing = \the\@tempdima}%
\@tempdima=\fontdimen1\font%
% \typeout{slant per point = \the\@tempdima}%
% now apply fudge factor:
\@tempdima=0.1\@tempdima%
\@tempdima=\EF@slantcorr\@tempdima%
\@tempdima=0.1\@tempdima%
% (the eccentric multiplication is to reduce rounding error and
% reduce the risk of the dimen getting too big)
\edef\@tempcmda{\strip@pt\@tempdima}%
% \typeout{fudged slant per point/points = \@tempcmda}%
% \@tempdima=#1%
\@tempdima=#2% Changed from width #1 to height #2
\@tempdima=\@tempcmda\@tempdima%
% \typeout{Width of C = #1}%
```

```
% \typeout{Height of C = #2}%
% \typeout{slant in points (for C) = \the\@tempdima}%
\@tempdima=0.5\@tempdima%
% \typeout{half slant in points (for C) = \the\@tempdima}%
% % now apply fudge factor:
% \@tempdima=0.1\@tempdima%
% \@tempdima=\EF@slantcorr\@tempdima%
% \@tempdima=0.1\@tempdima%
% % (the eccentric multiplication is to reduce rounding error and
% % reduce the risk of the dimen getting too big)
\edef\EF@slantC{\the\@tempdima}%
% \typeout{final half slant in points (for C) = \EF@slantC}%
%
% \EF@slantC is now 1/2 the slant of a C;
% need this for shifting back and forth
%
% Now calculate horizontal offset between rules
% \@tempcmda is still the fudged slant per point/points, so...
\@tempdimb=#8% centre line spacing of rules
\@tempdimb=\@tempcmda\@tempdimb%
\@tempdimb=0.5\@tempdimb%
\edef\EF@rulehoffset{\the\@tempdimb}%
%
% below is the old, mad, and incorrect (?) calc
%
% \begin{madness}
%
% \@tempdima=#2
% \@tempdima=100\@tempdima
% \advance\@tempdima by0.5pt\@settopoint\@tempdima
% \edef\@tempcmda{\strip@pt\@tempdima}%
% \@tempdima=100pt%
% \divide\@tempdima by\@tempcmda%
% \edef\@tempcmda{\strip@pt\@tempdima}%
% \@tempdima=#8%
% \@tempdima=\@tempcmda\@tempdima%
% \@tempdimb=\EF@slantC%
% \@tempdimb=2\@tempdimb% since \EF@slantC is actually 1/2 slantC
% \edef\@tempcmda{\strip@pt\@tempdimb}%
% \@tempdima=\@tempcmda\@tempdima%
% \@tempdima=0.5\@tempdima%
% %
% \edef\EF@rulehoffset{\the\@tempdima}%
%
% \end{madness}
%
% \typeout{\protect\EF@rulehoffset\space = \EF@rulehoffset}%
%
% \EF@ruleoffset is now 1/2 the horizontal offset between rules;
% need this for shifting back and forth even more.
%
\@tempdima=#1%              \
\@tempdima=0.01\@tempdima%  \
\@tempdima=#4\@tempdima%     } move back to add cross-strokes
```

```
\kern-\@tempdima%            /
\kern\EF@slantC%            /  Slant correction forward for italic
% lift = (height C - depth C - rule gap - rule thickness)/2  + offset
% Note that ‘rule gap’ is the distance between the centre lines of
% the rules, not the gap between them
\@tempdima=#2%                      \
\advance\@tempdima by-#3%            \
\advance\@tempdima by-#8%             \ calc lift of cross-strokes
\advance\@tempdima by-#7%             /
\@tempdima=0.5\@tempdima%            /
\advance\@tempdima by#9%            /
% 0.17ex was 1/2 height c/s; height 0.07ex; spacing 0.2ex
\@tempdimc=#8%             \ calc cross-stroke spacing
\advance\@tempdimc by-#7%  /
\raisebox{\@tempdima}{%
\vbox{%                             \
\@tempdimb=#1%                \      \
\@tempdimb=0.01\@tempdimb%     } calc width of cross-stroke
\@tempdimb=#5\@tempdimb%      /        \
\hbox{\kern\EF@rulehoffset%             \ top rule
\vbox{\hrule width\@tempdimb height#7}}% \
\nointerlineskip%                         } draw cross-strokes
\vskip\@tempdimc%                        / (using \@tempdima/b/c from above)
\@tempdimb=#1%                \        /
\@tempdimb=0.01\@tempdimb%     } calc width of cross-stroke
\@tempdimb=#6\@tempdimb%      /       /
\hbox{\kern-\EF@rulehoffset%         /
\vbox{\hrule width\@tempdimb height#7}}}}%  bottom rule
\@tempcnta=#4%                  \
\advance\@tempcnta by-#5%  (aha!) \
\@tempdima=#1%                     \
\@tempdima=0.01\@tempdima%          \ move forward to end of C
\@tempdima=\@tempcnta\@tempdima%    /
\kern\@tempdima%                   /
% shift  by slant amount          /
\kern-\EF@slantC%               /   slant correction
}
%
```

\EF@fakeeurobase This command contains the letter that’s used as the base to fake a euro symbol – it’s used by the \EFruleeuro command.

```
\providecommand\EF@fakeeurobase{C}
```

\EFruleeuro The internal command to generate faked euro symbols; meant to be used in config files and things if the user wants to create new sorts of fake euro symbols.

```
\def\EFruleeuro#1#2#3#4#5#6#7{{% Confine defs to \EFruleeuro
%             #1 = percentage of width of C that back shift of
%                  rules is; 110 usually
%             #2 = percentage of width of C that width of
%                  top rules is
%             #3 = percentage of width of C that width of
%                  bottom rule is
%             #4 = rule thickness (dimen)
%             #5 = rule spacing (dimen)
```

```
%              #6 = extra rule lift (dimen)
%              #7 = slant fudge factor (percentage); this is used to
%                   scale the slant per point dimension from the tfm
%                   file; it's often wrong in terms of the C glyph
\settowidth{\@tempdima}{\EF@fakeeurobase}%
\settoheight{\@tempdimb}{\EF@fakeeurobase}%
\settodepth{\@tempdimc}{\EF@fakeeurobase}%
\edef\EF@Cwidth{\the\@tempdima}%
\edef\EF@Cheight{\the\@tempdimb}%
\edef\EF@Cdepth{\the\@tempdimc}%
\advance\@tempdimb by\@tempdimc%
% \edef\totalCheight{\the\@tempdimb}%\totalCheight is height+depth of C
% \typeout{\protect\EF@Cwidth\space = \EF@Cwidth}%
% \typeout{\protect\EF@Cheight\space = \EF@Cheight}%
% \typeout{\protect\EF@Cdepth\space = \EF@Cdepth}%
% First job: deal with slant fudge
\def\EF@slantcorr{#7}%
\ifx\EF@slantcorr\@empty\def\EF@slantcorr{100}\fi%
% Ensure that \EF@slantcorr is 100 (%) if not specified.  This
% parameter is used by \EF@crossstrokes
\def\EF@backshift{#1}% percentage of \EF@Cwidth that back shift is
\@tempdimb=0.01\@tempdima%          swapped these two lines;
\@tempdimb=\EF@backshift\@tempdimb% dimen can get too big otherwise
% \@tempdimb now = length of shift back to add cross-strokes
% Note that this ignores the italic corrections.  Is this right?
% (ignoring italic correction seems to be right; but...  The
% introduction of the horizontal offset to the rules might not be
% right to ignore; it seems suitable in tests, but?  Since the amount
% should always be small, I'll ignore it for now - dealing with it is
% awkward and the offset *should* help things fit together better.)
% \@tempdima now = width of C
% typeset the faked euro
%
% Assumption: that the cross-strokes don't extend to the right of the
% C (allows the width calculations to be a bit simpler than they
% otherwise would be).
%
% \@tempdima = width of C; \@tempdimb = \EF@backshift x width of C
% \typeout{\the\@tempdima\space = width of C}%
% \typeout{\the\@tempdimb\space = \protect\EF@backshift x width of C}%
% Following line: make box = largest of (\@tempdima,\@tempdimb)
\ifdim\@tempdimb>\@tempdima\@tempdima=\@tempdimb\fi%
\makebox[\@tempdima][r]%
{\EF@fakeeurobase\EF@crossstrokes{\EF@Cwidth}{\EF@Cheight}{\EF@Cdepth}%
                                  {\EF@backshift}{#2}{#3}{#4}{#5}{#6}}%
}}
```

\SelectOnWeight

```
if the current fount series:
     is in the  light     list, do #1; else
if it's in the  medium    list, do #2; else
if it's in the  bold      list, do #3; else
if it's in the  ultrabold list, do #4; else
do #2
```

```
\def\SelectOnWeight#1#2#3#4{%
```

```
\EF@checkiflisted{\f@series}{\EFlightserieslist}%
\ifEF@listed\EF@debugrep{SelectOnWeight light}#1%
\else
\EF@checkiflisted{\f@series}{\EFmediumserieslist}%
\ifEF@listed\EF@debugrep{SelectOnWeight medium}#2%
\else
\EF@checkiflisted{\f@series}{\EFboldserieslist}%
\ifEF@listed\EF@debugrep{SelectOnWeight bold}#3%
\else
\EF@checkiflisted{\f@series}{\EFultraboldserieslist}%
\ifEF@listed\EF@debugrep{SelectOnWeight ultrabold}#4%
\else\EF@debugrep{SelectOnWeight default to medium}%
#2%
\fi\fi\fi\fi}
```

`\mediumruleeuronorm`
`\mediumruleeuronoslant`
`\mediumruleeurobigslant`

This is the standard command for faking a euro symbol. It selects one of four weights of cross-stroke depending on the current fount series.

The light and ultra bold weights are fairly arbitrary.

```
\providecommand*{\mediumruleeuronorm}[2]{\EF@debugrep{medium rule euro norm}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{72}{0.07ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{72}{0.14ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{72}{0.18ex}{0.27ex}{#1}{#2}}% ultra bold - new numbers
}
\providecommand*{\mediumruleeuronoslant}[2]{\EF@debugrep{medium rule euro noslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{80}{0.04ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{80}{0.07ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{80}{0.14ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{80}{0.18ex}{0.27ex}{#1}{#2}}% ultra bold
}
\providecommand*{\mediumruleeurobigslant}[2]{\EF@debugrep{medium rule euro bigslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{60}{0.04ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{60}{0.07ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{60}{0.14ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{60}{0.18ex}{0.27ex}{#1}{#2}}% ultra bold
}
```

`\lightruleeuronorm`
`\lightruleeuronoslant`
`\lightruleeurobigslant`

A complementary command for faking a euro with lighter cross-strokes; suitable for founts like Computer Modern Roman.

```
\providecommand*{\lightruleeuronorm}[2]{\EF@debugrep{light rule euro norm}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{72}{0.02ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{72}{0.04ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{72}{0.07ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{72}{0.14ex}{0.27ex}{#1}{#2}}% ultra bold
}
\providecommand*{\lightruleeuronoslant}[2]{\EF@debugrep{light rule euro noslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{80}{0.02ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{80}{0.04ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{80}{0.07ex}{0.27ex}{#1}{#2}}% bold
```

```
{\EFruleeuro{110}{80}{80}{0.14ex}{0.27ex}{#1}{#2}}% ultra bold
}
\providecommand*{\lightruleeurobigslant}[2]{\EF@debugrep{light rule euro bigslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{60}{0.02ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{60}{0.04ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{60}{0.07ex}{0.27ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{60}{0.14ex}{0.27ex}{#1}{#2}}% ultra bold
}
```

\heavyruleeuronorm
\heavyruleeuronoslant
\heavyruleeurobigslant

A complementary command for faking a euro with heavy cross-strokes; it seemed inevitable after the other two. I'm not sure that there's any particular sense to the bold and ultra bold values in this case.

I'm still not *completely* happy with this, but it's much better now.

```
\providecommand*{\heavyruleeuronorm}[2]{\EF@debugrep{heavy rule euro norm}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{72}{0.14ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{72}{0.18ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{72}{0.22ex}{0.30ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{72}{0.26ex}{0.33ex}{#1}{#2}}% ultra bold
}
\providecommand*{\heavyruleeuronoslant}[2]{\EF@debugrep{heavy rule euro noslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{80}{0.14ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{80}{0.18ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{80}{0.22ex}{0.30ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{80}{0.26ex}{0.33ex}{#1}{#2}}% ultra bold
}
\providecommand*{\heavyruleeurobigslant}[2]{\EF@debugrep{heavy rule euro bigslant}%
\SelectOnWeight%
{\EFruleeuro{110}{80}{60}{0.14ex}{0.27ex}{#1}{#2}}% light
{\EFruleeuro{110}{80}{60}{0.18ex}{0.27ex}{#1}{#2}}% medium
{\EFruleeuro{110}{80}{60}{0.22ex}{0.30ex}{#1}{#2}}% bold
{\EFruleeuro{110}{80}{60}{0.26ex}{0.33ex}{#1}{#2}}% ultra bold
}
```

8.5 Code to use marvosym

Code to use marvosym fount; needs faked italic tfm (and dvi driver able to fake an italic)

\EF@mvs

```
\def\EF@mvs{\fontencoding{U}\fontfamily{zmvs}\fontseries{m}\selectfont}
```

Note that the `\mbox` is now redundant; leave it in in case `\EF@pmb` changes again

\marvosymeuro

```
\providecommand{\marvosymeuro}[1]{%
\EF@debugrep{marvosymeuro}%
\SelectOnWeight%
{\EF@mvs\char#1}% light
{\EF@mvs\char#1}% medium
```

```
{\mbox{\EF@pmsb{\EF@mvs\char#1}}}% bold
{\mbox{\EF@pmsb{\EF@mvs\char#1}}}% ultra bold
}
```

\marvosymserifeuro
\marvosymsanseuro
\marvosymmonoeuro

```
\providecommand
 {\marvosymserifeuro}{\EF@debugrep{marvosymserifeuro}\marvosymeuro{101}}
\providecommand
 {\marvosymsanseuro}{\EF@debugrep{marvosymsanseuro}\marvosymeuro{99}}
\providecommand
 {\marvosymmonoeuro}{\EF@debugrep{marvosymmonoeuro}\marvosymeuro{100}}
```

8.6 Faking a bold character

Original comments: `\EF@pmb` fakes bold; modified from the TEXbook

I don't pretend to understand it all, but this seems to do the job better, and ensures that the width taken by the box is the width taken by the printed glyphs. This is used by `\marvosymeuro` Original code:

```
\def\EF@pmb#1{\makebox{\setbox\@tempboxa=\hbox{#1}%
  \kern0em\copy\@tempboxa\kern-\wd\@tempboxa
  \kern.025em\raise.0216em\copy\@tempboxa\kern-\wd\@tempboxa
  \kern.025em\box\@tempboxa }}
```

This code produces a better faked bold; the idea of using a hexagonal arrangement was suggested by Donald Arseneau. You can probably tell I wrote the `\EF@pmb` macro myself :-) I might move to an octagonal arrangement later on.

Marvosym complicates matters; a vertical shift of `0.866 x 0.020em` makes the horizontal strokes collide when you fake a bold Marvosym euro symbol. The solution is to move away from a regular hexagon and use a smaller vertical shift.

So now there's two commands: `\EF@pmb` for normal fake bold, and `\EF@pmsb` (semi bold) for the Marvosym euro symbols. It's only slightly less heavy than the usual version.

\EF@pmbshift
\EF@pmb
\EF@pmsb

```
\newlength{\EF@pmbshift}
%
\newcommand{\EF@pmsb}[1]{%
\EF@debugrep{EF@pmsb}%
\EF@pmbshift=0.020em% as was
\hbox{%
\rlap{#1}%
\kern0.5\EF@pmbshift%
\raisebox{0.50\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\raisebox{-0.50\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\kern1\EF@pmbshift%
\raisebox{0.50\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\raisebox{-0.50\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\kern0.5\EF@pmbshift%
#1}}
%
\newcommand{\EF@pmb}[1]{%
\EF@debugrep{EF@pmb}%
```

```
\EF@pmbshift=0.020em% as was
\hbox{%
\rlap{#1}%
\kern0.5\EF@pmbshift%
\raisebox{0.866\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\raisebox{-0.866\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\kern1\EF@pmbshift%
\raisebox{0.866\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\raisebox{-0.866\EF@pmbshift}[0pt][0pt]{\rlap{#1}}%
\kern0.5\EF@pmbshift%
#1}}
```

8.7 The code to select and print euro symbols

But better yet: add a pair of options: '`adobeeurofont`' or '`marvosym`'. The `marvosym` option will set up `\monoeuro` (etc) to use marvosym glyphs; the `adobeeurofont` option (default) will set `\monoeuro` (etc) to use adobe glyphs.

Do this, *and* add documentation on fiddling with `\userlist` (etc) so you can use both

`\texteuro` is a problem: this command is here to help.

```
if <fam> is in \faketexteurolist, #2,
  else
  if \texteuro is defined and TS1<fam>.fd exists #1,
    else #2
    fi fi
```

`\EFiftexteuroexists`

```
\def\EFiftexteuroexists#1#2{%
\EF@checkiflisted{\f@family}{\faketexteurolist}%
\ifEF@listed#2%
\else
\ifx\texteuro\@undefined#2% if \texteuro doesn't exist, #2 and finish
\else%
\IfFileExists{ts1\f@family.fd}%
{#1}% if ts1<fam>.fd exists, #1.  Can't usefully test for glyph existing
{#2}% if ts1<fam>.fd doesn't exist, #2 and finish
\fi\fi}
```

`\zpeutteuro` `\zpeureuro` `\zpeusseuro`

Define these for `\monoeuro` etc., to use. Note that the command names are such that if you happen to be in `zpeutt` and say `\euro`, you'll get `\zpeutteuro` executed straight away. I don't have to list these fount families in `\seriflist` etc.

```
\providecommand{\zpeutteuro}{\EF@debugrep{zpeutteuro}%
{\fontencoding{U}\fontfamily{zpeutt}\selectfont e}}
\providecommand{\zpeureuro}{\EF@debugrep{zpeureuro}%
{\fontencoding{U}\fontfamily{zpeur}\selectfont e}}
\providecommand{\zpeusseuro}{\EF@debugrep{zpeusseuro}%
{\fontencoding{U}\fontfamily{zpeuss}\selectfont e}}
```

Use either marvosym or Adobe's Eurofonts for serif/sans/mono lists

These might have already been defined in the cfg file; don't want to over-ride those definitions, so `\providecommand` them.

It's not safe to assume that either Adobe's Eurofonts or the Marvosym fount are in fact installed, but I don't see any easy way of finding out the truth of the matter. I suspect it's best to leave it up to the user to get the configuration right. Careful with that documentation, Eugene.

\monoeuro
\serifeuro
\sanseuro

```
\ifEF@marvosym
\providecommand{\monoeuro}{\EF@debugrep{monoeuro}{\marvosymmonoeuro}}
\providecommand{\serifeuro}{\EF@debugrep{serifeuro}{\marvosymserifeuro}}
\providecommand{\sanseuro}{\EF@debugrep{sanseuro}{\marvosymsanseuro}}
\else
\providecommand{\monoeuro}{\EF@debugrep{monoeuro}\zpeutteuro}
\providecommand{\serifeuro}{\EF@debugrep{serifeuro}\zpeureuro}
\providecommand{\sanseuro}{\EF@debugrep{sanseuro}\zpeusseuro}
\fi
```

Slightly more involved code for \makeusereuro: make it issue a warning if it's not user defined.

\makeusereuro

```
\providecommand{\makeusereuro}
               {\EF@debugrep{makeusereuro}\makefakemediumeuro%
%
\PackageWarningNoLine{eurofont}{%
You have tried to use the \protect\makeusereuro\space command
to\MessageBreak print a euro symbol, but you have not defined
the\MessageBreak \protect\makeusereuro\space command. This has
probably happened in\MessageBreak the \protect\euro\space command. I
shall print a faked euro symbol\MessageBreak for now}}
```

\chinaeeuro

I have some misgivings about faking a bold from this euro symbol; it's a beautiful glyph which doesn't really deserve this sort of abuse.

Removed the anomalous parameter from definition of \chinaeeuro. What was I thinking of?

And finally… Test for the existence of china2e.sty, and assume that China2e's not been installed if the package file is missing. I'd rather test for the existence of the tfm file, but I don't know how: if you know, I'd appreciate an email explaining how.

```
\DeclareFontFamily{OT1}{chin}{}
\DeclareFontShape{OT1}{chin}{m}{n}{<-> china10}{}
%
\providecommand{\chinaeeuro}{%
\IfFileExists{china2e.sty}%
{\EF@debugrep{chinaeeuro}%
\SelectOnWeight%
{{\usefont{OT1}{chin}{m}{n}\char255}}% light
{{\usefont{OT1}{chin}{m}{n}\char255}}% medium
{\mbox{\EF@pmb{\usefont{OT1}{chin}{m}{n}\char255}}}% bold
{\mbox{\EF@pmb{\usefont{OT1}{chin}{m}{n}\char255}}}% ultra bold
}%
{\EF@debugrep{chinaeeuro - we have a problem}%
\PackageWarning{eurofont}{%
The \protect\euro\space command
```

```
is trying to print a euro symbol\MessageBreak
from the China2e fount, but I can't find the file\MessageBreak
china2e.sty.\MessageBreak\MessageBreak
I'm assuming that you've not got the China2e font\MessageBreak
installed, so I'm printing a faked euro instead\MessageBreak}%
\makefakeeuro}}
```

\cmrfakeeuro
\cmssfakeeuro
\cmttfakeeuro

```
\providecommand{\cmrfakeeuro}{%
\EF@debugrep{\protect\cmrfakeeuro}\fakelighteuro}
%
\providecommand{\cmssfakeeuro}{%
\EF@debugrep{\protect\cmssfakeeuro}\fakemediumeuro}
%
\providecommand{\cmttfakeeuro}{%
\EF@debugrep{\protect\cmttfakeeuro}\fakemediumeuro}
```

\makefakeeuro
\makefakeheavyeuro
\makefakemediumeuro
\makefakelighteuro

If `\<fam>fakeeuro` exists, do `\<fam>fakeeuro`; else `\fakelighteuro`, `\fakemediumeuro`, or `\fakeheavyeuro` depending).

`\providecommand`s again so that these definitions can be over-ridden in the config file

```
\providecommand{\makefakemediumeuro}{\EF@debugrep{makefakemediumeuro}%
\@ifundefined{\f@family fakeeuro}{\fakemediumeuro}%
{\csname\f@family fakeeuro\endcsname}}
%
\providecommand{\makefakelighteuro}{\EF@debugrep{makefakelighteuro}%
\@ifundefined{\f@family fakeeuro}{\fakelighteuro}%
{\csname\f@family fakeeuro\endcsname}}
%
\providecommand{\makefakeheavyeuro}{\EF@debugrep{makefakeheavyeuro}%
\@ifundefined{\f@family fakeeuro}{\fakeheavyeuro}%
{\csname\f@family fakeeuro\endcsname}}
%
\providecommand{\makefakeeuro}{\EF@debugrep{makefakeeuro}%
\@ifundefined{\f@family fakeeuro}{%
% if \<fam>fakeeuro doesn't exist, do this:
\EF@checkiflisted{\f@family}{\fakelightlist}\relax
  \ifEF@listed \makefakelighteuro
    \else
    \EF@checkiflisted{\f@family}{\fakemediumlist}\relax
    \ifEF@listed \makefakemediumeuro
      \else
      \EF@checkiflisted{\f@family}{\fakeheavylist}\relax
      \ifEF@listed \makefakeheavyeuro
        \else%                If <fam>'s not listed,
        \makefakemediumeuro%  print medium faked euro
        \fi \fi \fi
}%
% If \<fam>fakeeuro does exist, execute it:
{\csname\f@family fakeeuro\endcsname}}
```

This turned into a heavy bummer. Re-wrote the faking code to push the top rule to the right and the bottom rule to the left when the fount slants.

\pzcfakeeuro

```
\providecommand{\pzcfakeeuro}{\EF@debugrep{pzcfakeeuro}%
% \@tempdima=1ex% you what?
% commented out until I find out what I thought I was doing
\fakemediumeuro[-0.1ex]}
```

8.8 Make euro commands

Can't just `\let` one equal to the other because of the debugging reports.

\maketexteuro
\cmeuro

```
\providecommand{\maketexteuro}{\EF@debugrep{maketexteuro}%
\EFiftexteuroexists{\texteuro}{\makefakeeuro}}
%
\providecommand{\cmeuro}{\EF@debugrep{cmeuro}%
\EFiftexteuroexists{\texteuro}{\makefakeeuro}}
```

If the eurosym option's been specified, switch to eurosym's version of the euro for: the Computer Modern families, serifeuro, sanseuro, and monoeuro, and the default euro.

This is the only place where these euro commands are defined; use `\providecommand` so they can be over-ridden by the config file.

\makecmeuro
\makedefaulteuro
\makeserifeuro
\makesanseuro
\makemonoeuro

```
\ifEF@eurosym
\EF@debugrep{Using eurosym's euro command}
\providecommand{\makecmeuro}     {\EF@debugrep{makecmeuro;      ESeuro}\ESeuro}
\providecommand{\makedefaulteuro}{\EF@debugrep{makedefaulteuro; ESeuro}\ESeuro}
\providecommand{\makeserifeuro}  {\EF@debugrep{makeserifeuro;   ESeuro}\ESeuro}
\providecommand{\makesanseuro}   {\EF@debugrep{makesanseuro;    ESeuro}\ESeuro}
\providecommand{\makemonoeuro}   {\EF@debugrep{makemonoeuro;    ESeuro}\ESeuro}
\else
\EF@debugrep{Not using eurosym's euro command}
\providecommand{\makecmeuro}     {\EF@debugrep{makecmeuro     }\cmeuro}
\providecommand{\makedefaulteuro}{\EF@debugrep{makedefaulteuro}\makefakeeuro}
\providecommand{\makeserifeuro}  {\EF@debugrep{makeserifeuro  }\serifeuro}
\providecommand{\makesanseuro}   {\EF@debugrep{makesanseuro   }\sanseuro}
\providecommand{\makemonoeuro}   {\EF@debugrep{makemonoeuro   }\monoeuro}
\fi
```

\makechinaeeuro This command needs a test to detect the existence of the china2e tfm file before trying to use the fount

```
\providecommand{\makechinaeeuro}
               {\EF@debugrep{makechinaeeuro}\chinaeeuro}
```

Note: `\makecmeuro`+`\maketexteuro` could have identical definitions, since `\makefakeeuro` will call `\<fam>euro` if it exists before trying the generic fake euro code. (much later) Does anyone know what I mean by this?

```
If \f@family is in \userlist, \makeusereuro
  else
  if \<fam>euro exists, \<fam>euro
    else
```

```
if \f@family is in \texteurolist, \maketexteuro
  else
  if \@family is in \chinaelist, \makechinaeeuro
    else
    if \f@family is in \cmlist, \makecmeuro
      else
      if \f@family is in \serifeurolist, \makeserifeuro
        else
        if \f@family is in \sanseurolist, \makesanseuro
          else
          if \f@family is in \monoeurolist, \makemonoeuro
            else
            if \f@family is in \makefakemediumeurolist, \makefakemediumeuro
              else
              if \f@family is in \makefakeheavyeurolist, \makefakeheavyeuro
                else
                if \f@family is in \makefakelighteurolist, \makefakelighteuro
                  else \makedefaulteuro
                  fi fi fi fi fi fi fi fi fi fi fi
```

`\makecmeuro`:

```
\texteuro if:
   not in \faketexteurolist and  TS1<fam>.fd and \texteuro exist,
      \makefakeeuro otherwise
```

`\maketexteuro`:

```
\texteuro if:
   not in \faketexteurolist and  TS1<fam>.fd and \texteuro exist,
  \makefakeeuro otherwise
```

This is it… The `\relax`es are in there just to suppress unwanted spaces and let me use the visual formatting in the command that I wanted to.

`\EFeuro`

```
\DeclareRobustCommand{\EFeuro}{\EF@debugrep{start EFeuro:
 \f@encoding/\f@family/\f@series/\f@shape}%
 \EF@checkiflisted{\f@family}{\userlist}\relax
 \ifEF@listed\EF@debugrep{EFeuro makeuser euro}\makeusereuro
  \else
  \@ifundefined{\f@family euro}{\@tempswatrue}%   test for \<fam>euro
  {\EF@debugrep{EFeuro \f@family euro}\relax%           do \<fam>euro
  \@tempswafalse\csname\f@family euro\endcsname}\relax% do \<fam>euro
  \if@tempswa% if \<fam>euro doesn't exist, do the next test
    \EF@checkiflisted{\f@family}{\texteurolist}\relax
    \ifEF@listed \EF@debugrep{EFeuro texteuro}\maketexteuro
      \else
      \EF@checkiflisted{\f@family}{\chinaelist}\relax
      \ifEF@listed \EF@debugrep{EFeuro China2e euro}\makechinaeeuro
        \else
        \EF@checkiflisted{\f@family}{\cmlist}\relax
        \ifEF@listed \EF@debugrep{EFeuro cm euro}\makecmeuro
          \else
          \EF@checkiflisted{\f@family}{\seriflist}\relax
```

```
          \ifEF@listed \EF@debugrep{EFeuro serif euro}\makeserifeuro
            \else
            \EF@checkiflisted{\f@family}{\sanslist}\relax
            \ifEF@listed \EF@debugrep{EFeuro sans euro}\makesanseuro
              \else
              \EF@checkiflisted{\f@family}{\monolist}\relax
              \ifEF@listed \EF@debugrep{EFeuro mono euro}\makemonoeuro
                \else
                \EF@checkiflisted{\f@family}{\fakelightlist}\relax
                \ifEF@listed \EF@debugrep{EFeuro fake light euro}\makefakelighteuro
                  \else
                  \EF@checkiflisted{\f@family}{\fakemediumlist}\relax
                  \ifEF@listed \EF@debugrep{EFeuro fake med euro}\makefakemediumeuro
                    \else
                    \EF@checkiflisted{\f@family}{\fakeheavylist}\relax
                    \ifEF@listed \EF@debugrep{EFeuro fake heavy euro}\makefakeheavyeuro
                      \else \EF@debugrep{EFeuro default euro}\makedefaulteuro
                      \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi}
```

Now then. . . eurofont's euro-generating command is defined as `\EFeuro` to begin with. That's so that it can be defined without clashing with anything else (with luck). The following code is to try and ensure that the `\euro` command used in the document is what eurofont wants it to be. If eurosym's been loaded, its `\euro` command is saved as `\ESeuro` just after its been loaded, and if any command called `\euro` exists at all, it's renamed `\oldeuro` at the `\begin{document}` command. Yes, that does mean that `\ESeuro` and `\oldeuro` can end up identical if you've loaded the eurosym package.

```
\AtBeginDocument{%
% \@ifpackageloaded{eurosym}{\let\ESeuro\euro}{}% save eurosym's \euro
% No: this is redundant to an extent.  If the eurosym option has been
% specified, \ESeuro has already be \let to \euro.  Best to leave it
% at that, I think: any already-defined \euro command will be saved
% as \EFoldeuro anyway.
\let\oldeuro\euro% save the previous \euro command (if one exists)
\let\euro\EFeuro}% make the \euro command be \EFeuro come what may
```

```
\endinput
```