

The `universal` Font

Version 2.0

Christian Holm*

August 1 1998

Abstract

This is my implementation of Herbert Bayer’s “universal” font in METAFONT for `TEX` and `LATEX`. Extensive support for `LATEX` is supplied. This font is in no way intended to be a correct, not to mention a complete implementation of Herbert Bayer’s original design. This document describes how to use the font with `LATEX`, and also the source code for the characters of the font.

Contents

1	Introduction	2
2	The Font and it’s History	3
3	This METAFONT Implementation	5
3.1	The METAFONT version versus the Original	5
3.2	Features of the font	6
3.2.1	Series, Shapes, Sizes, and Special Characters	6
3.2.2	File Names for the METAFONT files	7
3.3	The <code>L^AT_EX</code> and NFSS Support	8
3.3.1	Options to <code>uni</code>	9
3.3.2	Font Selection Commands	9
3.3.3	Special Character Commands	11
3.3.4	Other Commands	11
3.3.5	File Names for the <code>L^AT_EX</code> files	12
3.4	The Major differences between version 1.0 and 2.0	12
4	<code>L^AT_EX</code> Support Files	13
4.1	The style file — <code>uni.sty</code>	13
4.1.1	Intilalizing	13
4.1.2	Options	14
4.1.3	Special Characters	14
4.1.4	Font Selection	17
4.2	The Font Definition Files — <code>*uni.fd</code>	20

*Niels Bohr Institute of Physics, Institute of Philosophy, Rethorics, and Education; University of Copenhagen; Denmark; E-mail: <cholm@nbi.dk>

5	The Font Macros and Programs	24
5.1	Base file — <code>unibase.mf</code>	24
5.1.1	Font macros	25
5.1.2	Control macroes	29
5.1.3	Drawing macroes	30
5.2	The Minuscles (lowercase letters) — <code>unilow.mf</code>	33
5.3	The Majuscles (uppercase letters) — <code>uniupp.mf</code>	40
5.4	The Numbers — (<code>unidig.mf</code>)	46
5.5	Specials — <code>unispe.mf</code>	49
5.6	Punctuations — <code>unipun.mf</code>	51
5.7	Accents — <code>uniacc.mf</code>	55
5.8	Ligatures — <code>unilig.mf</code>	57
5.9	Extras — <code>uniext.mf</code>	59
6	Font driver files	66
7	Commands for the Documentation — <code>unidoc.sty</code>	67
7.1	Initializing	67
7.2	Macros for the Index	68
7.3	Descriptive Environments	69
7.4	Useful Commands	70
7.4.1	The Font Charts	71
7.5	Parameters and Miscillaneous Commands	73
A	Solution to the <code>\bauhausforms</code> problem	75
A.1	The Problem	75
A.2	Joseph Collins’ Solution	76
B	Copyrights — GNU General Public Lisence	77
C	Wishlist	77
D	Font Charts	79

1 Introduction

This package contains the METAFONT source and driver files for the “universal” font, designed by Herbert Bayer, a teacher at the Bauhaus school in Weimar, plus a L^AT_EX package to utilize this font, along with a number of font definition files, as required by the New Font Selection Scheme (NFSS).

About the Documentation

The full documentation of this font is rather large, more then 80 pages actually. Most of it is the programs for the characters for this font, so if you don’t know the METAFONT language, or don’t care how the charcters are created, you should insert `\OnlyDescription` into the preamble of `uni.dtx`.

Notice, that METAFONT macros are *not* indexed in this documentation. This is because it would take a *major* rewrite of the `doc` package to do so, and I really didn’t want to do that.

Some of the macros of `doc` has been redefined, and if you want to see which, or how I generally did the documentation, please refer to appendix 7.


If you in some way are unsatisfied with some of the characters of the font, do please read the documentation of that character. There may be some notes on why the character looks the way it does. Some of the comments, however are a bit silly and should be skipped at high speed (pretend there is a conditional that says `if not silly ... fi`).

This Version

This new version of the `universal` font, provides a number of new features and improvements, both to the font itself and to the `LATEX` support macros. Also, a number of corrections has been made to the font programs.

The reason why I jumped one whole version number from 1.0 to 2.0, is that I have taken a whole new approach to the font programs and shapes of the `universal` font, plus I decided to put some more effort into the `LATEX` support.

Thanks and Other Stuff

To those of you who have had the (mixed) pleasure of using version 1.0 of this font, one of the most noticable changes are to the symbol . In the old documentation, I complained that I couldn't find the exact solution to the problem this symbol posed. I also encouraged people to send me any solution they may have had — and guess what — somebody did! Therefore I would like to thank Joseph Collins for providing me with the solution. If you like Joseph Collins and I like Mathematical puzzles take a look in Appendix A to learn more about this problem and it's solution.

I also direct your attention to Appendix B for the copyright notice on the `universal` package (it's the *Gnu General Public Lisence* to those of you who know it).

If you in any way have gotten tempted to design your own font, or to implement some font into METAFONT, I feel obligated to bring you a warning, taken from the METAFONTbook by Donald E. Knuth:

Warning: Type design can be hazardous to your other interests. Once you get hooked, you will develop intense feelings about letterforms; the medium will intrude the message that you read. And you will perpetually be thinking of improvements to the fonts that you see everywhere, especially those of your own design.

2 The Font and it's History

Bauhaus

The Bauhaus school in germany originally located at Dessau, was a school for any kind of design, ranging from potery to furniture, from painting to — what was considered the prime form of design — architecture. Many famous designers came from, or taught at Bauhaus, for example Mies van der Rhoe, Herbert Bayer, Kandinsky, Walter Gropius and Gerrit Rietveld. The style “die stijl” was explored

here, and painters like Mondrian made large contributions to what today is known as “the Bauhaus style”.

The basic idea of the Bauhaus school, was to design items, which along with it’s aesthetic value, also had a high degree of functionality. Houses were meant to be suited for all kinds of living, while still keeping the beauty that make people glad to see thier house. Chairs should be comfortable for thier use, as well be able to fit-in in a normal house. All this should be done at a price that made it possible for everybody to own designer-furniture, houses, etc.

The political idea of Bauhaus, was that of a socialist one. Houses are for the people — they have to live in them, and that living should be a good as possible. Therefore the Bauhaus school saw it has its task to provide functional, beautiful everyday items that anybody could afford. Paintings and tapestry shouldn’t hang on museums or art galleries, but in peoples homes, where they would inrich thier everyday life.

Herbert Bayer and the “universal” Font

At Bauhaus typography was also studied, not just how written text should be typeset, and how printed characters should look like, but also what the essence of writting is, in it’s pratical and design-wise sense. This led Herbert Bayer to formulate some principles of writting:

experiment with simplified way of writingq:

1. this way of writingq is recommended by all typographic designers as the future way of writingq.
2. by writingq in minuscules our writingq looses nothingq, but is easier to read, considerably more economical.
3. why must you for one sound have two tokens, e.g. A and a? why two alphabets for one word, why this double set of signs, when the half is enough.

herbert bayer 1925

On this principles, Herbert Bayer designed a font, which should have no majuscles (upper case letters), easy to print, and easy to read since it didn’t have any unusefull decoration, but communicated the bare meaning of characters through the simplest forms needed to reconigise a character. This font he called “universal”.

This font contained abolutly no majuscles, since Bayer believed them to be superflous, as it is clear form the quote above.

At the time Herbert Bayer formulated these principles and designed the “universal” font, most printers used Gothic letters, which is allmost overly decorated, so his font ofcourse made contraversy.

Later on, in the 1930’ies, the Bauhaus school drew the attention of *Gestapo*¹ of Nazi² Germany. The school was finally closed in 1936 by Gestapo, because they believed they were promoters of Jewish and Communist culture and propaganda.

¹Gehemligche Stats Polizi

²Nationalsocialismus

In the aftermath of the closing of the school, most of its ideas were shunned by other designers, and the Bauhaus way of thinking died out. This is properly the reason why Herbert Bayers “universal” font is so little known today.

However, the “universal” font still stands as one of the most compelling developments in font designing. It represents an approach to designing where the functionality is as vital as the expression, and as such I believe it to be one of the most important fonts in the world today.

3 This METAFONT Implementation

This implementation of Herbert Bayers “universal” font, is not supposed to look *exactly* like the original design. Ofcourse I have tried to the best of my ability to mimic his design as far as I could. However, it is not an easy matter to find a complete, not to mention exact, sample or account of Bayers design.

This implementation is based on the samples I *could* find, and other implementations of the “universal” font. Many of these other implementations do differ from the original samples, and include characters I couldn’t find in any of the original. So whenever I found disparities, I mainly leaned on the original samples and my understanding of the original design.

3.1 The METAFONT version versus the Original

As mentioned above, Bayer never did design any majuscles for this font, but nonetheless, I have included them into this implementation. This I did, because I think most people will have a hard time writing in minuscles (lowercase letters) alone. Ofcourse, if you agree with Bayer, you should simply not use them.

There are also some other differences, mostly due to the fact, that I never found a complete sample of the original font. The major differences between the original font and this implementation is summarized below:

Majuscles: Majuscles are present, even though they weren’t in the original design.

Digits: These are based on other implementations, and my general conception of the original design.

Punctuations: As above.

Accents: As above.

Symbols: As above.

Bauhaus Symbols: I have added some various symbols I have found in connection with Bauhaus to the font. The reason is I find them beautiful, and I had some space to fill.

Numerous Shapes and Weights: I don’t think Bayer ever design slanted characters, a bold face version of the font, and he could never have design a small caps version of the font. However, these are present in this implementation. I included these features, because I believe them to be of general utility, and it makes the font conform more to the Computer Modern Roman font, and NFSS.

3.2 Features of the font

Rather than using `cmbase.mf`³, and then redefine some macros, I chose to make a new base file myself, i.e., `unibase.mf`. This file contains a number of macros⁴ I have used in the character programs.

The macros of `unibase.mf` actually reflects my conception of the font. There are three basic drawing macros:

`unicir` which draws a circle,

`uniarc` which draws a segment (arc) of a circle, and

`unilne` which draws a straight line.

I believe, that Bayer intended the font to be made of these two basic shapes: the arc, and the line. Also, to keep things simple, and therefore easy to print, all shapes should be of the same thickness, i.e., as if drawn with a pen of equal thickness. I have made one deviation from this, however. All majuscles are drawn with a thicker ‘pen’, which makes the output nicer, I think.

Incidentally, this made the programs of the characters much simpler, and shorter.

3.2.1 Series, Shapes, Sizes, and Special Characters

Change on
98/08/01,
Version 2.0

Below is a sample of each series/shape combination available in this implementation of the font, along with the \LaTeX commands that drive them:

Medium upright (`\textuni`): The dazed brown fox quickly gave 1234-567890 jumps!

Medium slanted (`\textunisl`): *The dazed brown fox quickly gave 1234-567890 jumps!*

Medium small caps (`\textunisc`): THE DAZED BROWN FOX QUICKLY GAVE 1234-567890 JUMPS!

Medium strict (`\textunist`): the dazed brown fox quickly gave 1234-567890 jumps!

Bold face upright (`\textunibf`): **The dazed brown fox quickly gave 1234-567890 jumps!**

Bold face slanted (`\textunibsl`): ***The dazed brown fox quickly gave 1234-567890 jumps!***

Bold face small caps (`\textunibsc`): **THE DAZED BROWN FOX QUICKLY GAVE 1234-567890 JUMPS!**

Bold face strict (`\textunibst`): **the dazed brown fox quickly gave 1234-567890 jumps!**

Everyone of these shapes are available in size 8, 9, 10, 12, 17 pt, and METAFONT can ofcourse create others.

Change on
98/08/01,
Version 2.0

³Computer Modern Roman base file

⁴A better name for ‘base’ would be ‘library’, and then the file would be `libuni.mf`, but to conform to CM, I used ‘base’.

Also a number of non-standard characters are available in the font. Below is a table of these characters along with the L^AT_EX commands that drive them.

Notice that “ and ” is present in the table. This is because these characters are not directly defined, but is supplied as *ligatures*. This can be done, because they are simple doubles of ‘ and ’.

A quick look on the table will also reveal some characters that generally isn't present in the standard OT1 encoding⁵, but generally present in the T1 encoding⁶. I have done this, both to provide an (almost) complete font for the European languages, but also because I anticipate the `universal` font some time in the future will shift, or at least be available, in the T1 encoding⁷. Please note, that in

■	<code>\bausquare</code>	●	<code>\baucircle</code>
▲	<code>\bautriangle</code>	⊕	<code>\bauhead</code>
⊕	<code>\bauforms</code>	ð	<code>\dh</code>
đ	<code>\dj</code>	ŋ	<code>\ng</code>
þ	<code>\th</code>	Œ	<code>\varQ</code>
Ð	<code>\DH</code>	Ɔ	<code>\DJ</code>
Œ	<code>\NG</code>	Ɔ	<code>\TH</code>
§	<code>\textsection</code> or <code>\S</code>	{	<code>\textbraceleft</code> or <code>\{</code>
}	<code>\textbraceright</code> or <code>\}</code>		<code>\textbar</code>
<	<code>\guilsinglleft</code>	>	<code>\guilsinglright</code>
«	<code>\guillemotleft</code>	»	<code>\guillemotright</code>
,	<code>\quotesinglbase</code>	,	<code>\quotedblbase</code>
‘	<code>\textquotedblleft</code>	’	<code>\textquotedblright</code>
ˆ	<code>\textogonek</code>	—	<code>\textunderscore</code>

Table 1: Non-standard characters in the universal font

the small caps shaped fonts, `\dh` does not give Ð, but a ð, that is a small caps shaped version of `\varQ`. Also there is no command `\varq` defined.

In appendix D is some charts showing the font in different series and shapes.

3.2.2 File Names for the METAFONT files

Base File and Source Files The base file and the files containing the code for the characters of the `universal` font, all starts with `uni`, to reflect the connection of the files. The next five possible letters reflects what kind of code is contained within the file, e.g., the base file ends in `base`, the file containing the code for the minuscules (lower case letters) end in `lower`, and so forth.

Font Driver Files The font driver filenames has been chosen to conform to the `fontname` scheme, because this scheme is used by most T_EX, L^AT_EX, and METAFONT systems (anyway those that use `kpathsea`, which is the most).

⁵The OT1 encoding is the 7 bit encoding of the Computer Modern fonts by Donald E. Knuth. 7 bit means it contain 128 (= 2⁷) characters.

⁶The T1 encoding is an encoding especially designed for the (western) European languages. It was founded by the T_EX User's Group, on a seminar in Cork, and is the basis of the `dc` fonts. T1 is an 8 bit encoding, which means it has 256 (= 2⁸) characters.

⁷T1 is generally considered *the* encoding of the future, and in the long term, it is most likely the encoding of L^AT_EX3.

The filename all contain the three characters `ful`, where the `f` stands for *public* and `ul` for *universal*.

Next comes a letter which is one of `m` (*medium*) or `b` (*bold*), which represents the series of the font.

Then comes one or two letters, which are `r` (*upright* or *roman*), `o` (*slanted*, or *oblique*), `c` (*small caps*), or `st` (*strict*), which represent the shape of the font.

Finally the filename ends with the designsizes in points.

Thus the complete syntax for the font driver file names is: The `fontname`

```

<filename> := <supplier><face><series><shape><size>.mf
<supplier> := f
<face>     := ul
<series>   := m | b
<shape>    := r | o | c | st
<size>     := | 8 | 9 | 10 | 12 | 17

```

scheme actually says to put `<encoding>` information after the `<shape>`, but since this is `8r` for `TEX` Text, it would make filenames longer than 8 characters in the cases of `<size>` of 10, 12, and 17, so this information is left out (which is permissible in `fontname`, but unfortunate).

This way of naming the font driver files will, if you use `kpathsea`, put the `ful*.pk` files in

```
<pk-base-dir>/public/universa/
```

and the `ful*.tfm` files in

```
<tfm-base-dir>/public/universa/
```

which I think is the intuitively correct place to put them. This also means, that the `*.mf` files provided with this package, should be placed in

```
<mf-source-base-dir>/public/universa/
```

again very intuitive.

Below is a table of the usual directory names under Unix-like and MSDOS-like (including Windows95) systems. `<mode>` is `dvips`'s name for your printer.

Varibale	Unix-like systems	MSDOS-like systems
<code><tfm-base-dir></code>	<code>/var/spool/texmf/pk/<mode>/</code>	<code>C:\FONTS\PK\<mode>\</code>
<code><tfm-base-dir></code>	<code>/var/spool/texmf/tfm/</code>	<code>C:\FONTS\TFM\</code>
<code><tfm-base-dir></code>	<code>/usr/local/lib/texmf/fonts/source/</code>	<code>C:\TEX\MFINPUTS\</code>

Table 2: Common directory names.

3.3 The `LATEX` and `NFSS` Support

To use the `universal` font with `LATEX 2ε`⁸, you should load the package `uni` with the command

```
\usepackage[<options>]{<uni>}
```

in your preamble (i.e., after `\documentclass` and before `\begin{document}`). `<options>` can be any of the options described below, but no other.

⁸I have made *no* attempt to provide support for `LATEX 2.09`, since this format is obsolete, and those who *do* use it, will properly never bother to look at CTAN for new fonts anyway.

3.3.1 Options to uni

`strict`
Change on
98/08/01,
Version 2.0

The `strict` option is intended to facilitate typesetting of the `universal` font in a *strict baubaus* fashion, that is *only* in minuscules.

In this font, only the series may be varied, that is, there is a bold series strict shaped font of any size, and a medium series strict shaped font of any size, in the `universal` family.

This option can be used in conjunction with options `medium` and `bold`. Please notice, that it doesn't make any sense to ask for a small caps or slanted shaped font, while using this option.

Notice that only `\textuni` and `\uni` is defined if option `strict` was given to `uni` package.

`default`
Change on
98/08/01,
Version 2.0

If you give the `default` option to the `uni` package, the default font of the document will be `universal`.

If you also used the option `bold` the default font will be the `universal` font in `bold` series. Otherwise it will be in medium series.

With this option, `\textit`, `\it`, and `\itshape` shifts to *universal slanted* font, i.e., there is no *italic* font available.

`\textcmr`
`\cmr`

To make it possible to change back to Donald E. Knuth's Computer Modern Roman font, even when the `default` option is given, we define macros `\textcmr` and `\cmr`, which switches the `\fontfamily` to `cmr` locally and globally respectively.

You should use this option with some care, since the `universal` font isn't very suited for longer texts, but rather for short letters, quotes, and other pieces of text where the graphical appearance is important.

`medium`
Change on
98/08/01,
Version 2.0
`bold`
Change on
98/08/01,
Version 2.0

When this option is given, command `\textuni` switches to `medium series universal` font, as do `\uni`. The other font selection commands behave as always (see below). This is the default option to `uni`, i.e., not normally needed.

If this option is given, commands `\textuni` and `\uni` switches to `bold series universal` font. Other font selection commands behave as always (see below). Notice that it makes no sense to give both option `medium` and option `bold` to the `uni` package.

3.3.2 Font Selection Commands

`\textuni`
`\uni`

These two commands only change the current font family to `uni` and *nothing else*. That means, that if you say for example

```
{\sl Hello \textuni{world}}
```

you get *both* 'Hello' and 'world' in slanted shape, and the output would be

Hello world

To put it in another way: Font encoding, shape, size, and baselineskip is preserved under `\textuni` and `\uni`, while font family is not.

Exceptions: If you gave the `bold` option to the `uni` package, then this command will always give you a bold series font. If you gave the `strict` option, then this command will always give you a strict shaped font.

These commands can be used in conjunction with \LaTeX commands `\textbf`, `\textsl`, and even `\sc`, or `\rm` to give different series and shapes.

`\textunirm`
`\unirm`

If you in the previous example intended to shift to medium upright uni-

versal font you could instead have used `\textunirm`, since this command *does not* preserve font shape, i.e., the shape is always changed to upright, regardless of the previous shape. So if you said

```
{\sl Hello \textunirm{world}}
```

you would get

Hello world

Actually `\textunirm` and `\unirm` isn't the only commands that aggressively changes most of the font parameters. `\textunis1`, `\textunisc`, and `\textunist`, *always* gives you *medium slanted*, `MEDIUM SMALL CAPS`, and `medium strict` respectively no matter what the values of `\f@shape` was before.

In the same category is `\textunibf`, `\textunibsl`, `\textunibsc`, and `\textunist` which always changes the font series to **bold**, along with change in shape (upright, slanted, small caps, and strict in that order).

All of the 'aggressive' commands, *do not* however change the *size* and *baselineskip* of the font. This should be done by using \LaTeX commands such as `\small`, `\Large`, `\fontsize{<size>}{<lineskip>}`, etc.

Notice that the 'aggressive' commands always changes to the appropriate font series. That is, even if you gave the **bold** option to `uni`, `\textunirm` will still give you *medium upright universal* font. This particular instance illustrates the use of the aggressive commands quite well I think.

To summarize: The 'aggressive' commands *doesn't* preserve family, series and shapes, but *does* preserve encoding, size, and baselineskip.

Warning: The font shifting commands `\textuni...` and `\uni...` in this section is *not* defined if you gave the `strict` option to the `uni` package.

One can also use the rather primitive command `\unifamily` in conjunction with `\selectfont` as described in *LaTeX 2_ε Font Selection*, to change the font family to **universal** if absolute control is preferred.

`\unifamily` is used by all the other font switching commands, so if you redefine it, or `\unifamilydefault`, you could get strange result.

This command normally expands to `uni`, which is the 'family' name of the **universal** package. If you redefine this command to be something else, e.g., `cmr`, `\unifamily` will load another font.

If the **bold** option to `uni` is used this command will select the default series of the **universal** font, defined in `\uniseriesdefault`, which ofcourse defaults to `b`, i.e., bold series. if you redefine `\uniseriesdefault` to be `m`, then `\uniseries` will select medium series fonts.

If **bold** option wasn't given, then this expands to nothing, as do `\uniseriesdefault`. This command is used by `\textuni` and `\uni`.

If the **strict** option to `uni` is used this command will select the default shape of the **universal** font, defined in `\unishapedefault`, which ofcourse defaults to `st`, i.e., strict shape. If you redefine `\unishapedefault` to be `n`, then `\uniseries` will select upright shaped fonts.

If **strict** option wasn't given, then this expands to nothing, as do `\unishapedefault`. This command is used by `\textuni` and `\uni`.

This command switches to *strict* shape, i.e., all majuscules will be typeset as minuscules. This makes it possible in a simple way to typeset text in the way Herbert Bayer thought it should, as evident from the citation above.

This command uses the command `\stdefault`, which defaults to `st`. If you redefine this to something else, e.g., `sl` you will get a *slanted* font.

This command is used by `\textunist`, `\textunibst`, `\unibst` and `\unist`, and is defined no matter what options you gave to the `uni` package.

Warning: Since *strict* is a non-standard shape, this command should not be used outside the `universal` font, since this may give you unexpected results.

3.3.3 Special Character Commands

For the individual commands that makes various special characters, please consult table 1 above.

3.3.4 Other Commands

`\k` The macro `\k` used in the `universal` font gives the accent ogenek, that is a reversed cidelia accent. It takes one argument, which should be a single letter, under which it puts the accent. For example, you could say `\textuni{\k{a}}` and get q .

`\DeclareUniChar` Now you can configure the special character commands of the font, via the commands `\DeclareUniChar` and `\DeclareUniCommand`. The commands defined via these commands will only work in accordance with its definition inside the `universal` font, and if defined elsewhere according to its definition there, else it will give an error message.

Change on 98/08/01, Version 2.0

`\DeclareUniChar` is used to define a command sequence representing a single character in the `universal` font, much like `\DeclareTextSymbol`, or in $\text{T}_{\text{E}}\text{X}$ `\chardef`, though the control sequence will produce an error message outside the `universal` font, and properle unexpected results outside the OT1 encoding⁹.

`\DeclareUniCommand` is used to define control sequences inside the `universal` font, representing many characters or doing complex maneuvers on characters and stuff. The optional argument to `\DeclareUniCommand` can be used to say how many arguments the control sequence should have, just like `\newcommand`. However, it is not possible to give a default first argument.

The below definition uses the `color` package to typeset a square, circle, and triangle in different colours¹⁰.

```
\DeclareUniCommand{\mybaufoms}{%
  \lower.5ex\hbox{\color{blue}\bautriangle}%
  \kern-.5em\raise.5ex\hbox{\color{red}\baucircle}%
  \kern-.5em\lower.5ex\hbox{\color{yellow}\bausquare}}
```

Yet another example, using arguments could be

```
\DeclareUniCommand{\mybaulogo}[1]{\bauhead\ {\Large #1}}
```

so you could say `\mybaulogo{Christian Holm}` and get:

 Christian Holm

⁹This should not be a problem.

¹⁰Since the documentation should be available to all, I can not provide you with the outcome of this example, since it needs the `color` package which may not be available on all sites. I suggest you try it out, or something similar, if you can.

and I bet you can come up with some even more useful and complex commands.

The syntax of `\DeclareUniChar` and `\DeclareUniCommand` is

```
\DeclareUniChar{⟨cmd⟩}{⟨slot⟩}  
\DeclareUniCommand{⟨cmd⟩}[⟨arg⟩]{⟨definition⟩}
```

where `⟨cmd⟩` is the user command defined, `⟨slot⟩` is the number of the character in the font, `⟨arg⟩` is the number of arguments and `⟨definition⟩` is what `⟨cmd⟩` does.

3.3.5 File Names for the L^AT_EX files

All the L^AT_EX files contain the three letters `uni`, to reflect the connectedness of the files. The font definition files all start with the letter code appropriate for the encoding.

To follow the scheme of the METAFONT files, it would be appropriate to place all L^AT_EX files in

```
⟨tex-base-dir⟩11/tex/latex/universa/
```

3.4 The Major differences between version 1.0 and 2.0

First of: a lot of bugs and errors has been corrected. In version 1.0, I had made the (stupid) mistake of calling the macro `mode_setup` before I defined the unsharped units. Ofcourse a quick look in the METAFONTbook showed be just how stupid this is. This made the font very vulnerable to mode specifications, which ofcourse isn't the idea.

Secondly: I chose a completely new approach to the character programs, which resultet in `unibase.mf`. The idea is to define a few macros, and then utilize those in the character programs, so that these programs can be kept simple, efficient, and intuitive.

A quick look at `unibase.mf` will also reveal that I chose a new way of adjusting the characters. This means that the macros `bauhaus...` present in version 1.0 no longer is needed, and since they only tended to ubscure things rather then simplify them, I went back to the plain METAFONT macro `beginchar`, which is much stronger.

All in all, `unibase.mf` provides a much stronger and uniform frame work for character design, then did the old `universal.mf`.

Thirdly: The file names have been kept inside MSDOS conventions, that is first name of maximum 8 charcaters, and last name of maximum 3 characters. This does mean, however, that some file names are not intuitive, but I have tried to make them as much as I could.

Also, every file associated with this font, except the font definition files (`*uni.fd`), and font driver files (`ful*.mf`) begins with the three letters `uni`, to emphasize the connection.

¹¹On Unix-like systems `⟨tex-base-dir⟩` is usually something like `/usr/local/lib/texmf/`, and on MSDOS-like systems something like `C:\TEX\`.

Fouthly: Some new font shapes are available, as explained above. I found out, during the design of the characters, that new the base file `unibase.mf` kept showing new potentiality, and the extension of the font to include more shapes was very easy inside the frame of this base, so I thought “What the heck!”

Fifthly: Some of the `bauhaus` symbols available in version 1.0, has been taken out, and some new, more general characters have been added. Most of the absent symbols were not really of general use, so I decided to take them out, since I was never really satisfied with those anyway. This also made the font contain exactly 128 characters, just like a normal Computer Modern Roman font.

Sixthly: I improved the \LaTeX and NFSS support considerably. The changes are legion, but let me sum up the most important here.

1. Stronger font selection commands.
2. Command names that should be more intuitive.
3. Preparations for T1 encoding.
4. Conformation to $\LaTeX 2_{\epsilon}$ style, and therefore a better chance to conform with the future $\LaTeX 3$ format.
5. More and better options.

4 \LaTeX Support Files

4.1 The style file — `uni.sty`

In version 2.0, the commands in this file has been redefined using the macros recommended in *$\LaTeX 2_{\epsilon}$ Font Selection* and *$\LaTeX 2_{\epsilon}$ for Class and Packages Writers*. This should make the commands and macros more portable, and secure. Further, it should make it upward-compatible with future releases of other packages and in the end $\LaTeX 3$.

4.1.1 Intilalizing

First we need to identify the package, its version and release date, etc.

```
1 \def\fileversion{v2.0}
2 \def\filedate{98/08/01}
3 \ProvidesPackage{uni}[\filedate\space\fileversion\space universal
4                               package.]
```

Then we setup some new `\if` commands, to help in different situations, depending on options passed to the package etc.

```
5 \newif\if@uni\@unifalse
6 \newif\ifstri@t\stri@tfalse
7 \newif\ifdef@ult\def@ultfalse
8 \newif\ifm@dium\m@diumfalse
9
```

4.1.2 Options

We define some options that can be passed to the package. Option `strict` is intended to make it possible to do *strict* Herbert Bayer typesetting, that is only in minuscles. Option `default` will make the *default* font `universal` of the entire document, while option `medium` and `bold` decides whether the default font used by `\textuni` is normal or bold series. Finally option `medium` is declared default.

```
10 \DeclareOption{strict}{\stri@ttrue}
11 \DeclareOption{default}{\def@ulttrue}
12 \DeclareOption{medium}{\m@diumtrue}
13 \DeclareOption{bold}{\m@diumfalse}
14 \ExecuteOptions{medium}
15 \ProcessOptions\relax
16
```

4.1.3 Special Characters

•¹²

•¹³

<code>\uni@init</code>	To facilitate the special character positions of the <code>universal</code> font, we define the
<code>\DeclareUniChar</code>	internal commands <code>\uni@init</code> , and the user commands <code>\DeclareUniChar</code> and
<code>\DeclareUniCommand</code>	<code>\DeclareUniCommand</code> .
<code>\Declare@Uni</code>	<code>\DeclareUniChar</code> and <code>\DeclareUniCommand</code> is very much like the L ^A T _E X com-
<code>\DeclareUni@xarg</code>	mands <code>\DeclareTextSymbol</code> and <code>\DeclareTextCommand</code> , except we have to take
<code>\DeclareUni@narg</code>	into account, that the commands may very well be defined in <code>ot1enc.def</code> or the
<code>\UniError</code>	like. Since some of the characters provided with this font, is usually part of an 8

bit font, I can't just use the character numbers straight of (this is a 7 bit font), and I have some very non-standard characters as well.

Though the two commands differ a bit, the main mechanism is the same, so I will explain them together, and note the differences¹⁴.

1. When either of the commands is used, we first call the macro `\Declare@uni`, which sets up 2 or 3 commands.
2. Inside `\Declare@Uni`, we define `\temp@a` to be `\uni@⟨cmd-name⟩`, where `⟨cmd-name⟩` is `⟨cmd⟩` stripped of the escape character (`\`).
3. Then we check whether `\cmd` is defined.
 - If `⟨cmd⟩` isn't defined, then we define `\temp@c` to give `\uni@⟨cmd-name⟩` (which isn't defined yet) inside the `universal` font, otherwise an error message (`\UniError`).
 - If `⟨cmd⟩` is defined, we define `\temp@b` to be `\no@uni@⟨cmd-name⟩`, `\temp@c` to give `\uni@⟨cmd-name⟩` inside the `universal` font, and `\no@uni@⟨cmd-name⟩` (which isn't defined yet) elsewhere. Then we define `\no@uni@⟨cmd-name⟩` to be what `⟨cmd⟩` currently is.

This finishes `\Declare@Uni` off.

¹²Change of `\DeclareUniChar` on 98/08/01, Version 2.0

¹³Change of `\DeclareUniCommand` on 98/08/01, Version 2.0

¹⁴The syntax of the macros are explained on page 10.

4. The next step depends a bit on which of the macros that is utilized. What is common is that `\uni@⟨cmd-name⟩` (hold in `\temp@a`) is defined to be the last argument of the command.
 - In `\DeclareUniChar`, we define `\uni@⟨cmd-name⟩` via the \TeX command `\chardef` to be `⟨slot⟩`, which should be a number (decimal, octal, or hexal) between 0 and 127. Using the \TeX primitive will make \LaTeX think of `⟨cmd⟩` as a single character.
 - If `\DeclareUniCommand` was given an optional argument, we use the command `\DeclareUni@xarg` to define `\uni@⟨cmd-name⟩` — which uses `\newcommand` with *one* optional argument — to be `⟨definition⟩`. Otherwise we use `\DeclareUni@narg` — which uses `\newcommand` with *no* optional arguments — to do the trick. Using the \LaTeX complex `\newcommand`, makes `⟨cmd⟩` a strong command.
5. Finally we define `⟨cmd⟩` to be `\temp@d`, which was the conditional command defined in step 2.

Notice the use of `\aftergroup`. This means that `⟨cmd-name⟩` shouldn't be too long, i.e., > 300 characters, which isn't a real limitation, but now I said it. This macro, together with the group in `\temp@c` is what made it possible to give arguments to `⟨cmd⟩` defined by `\DeclareUniCommand`.

This way of doing things has a number of benefits. 1) First of all makes it very simple to redefine the behaviour of the `universal` specific character commands. All you have to do is to use `\DeclareUniChar` or `\DeclareUniCommand` in your preamble (and *only* there). 2) Since \TeX are faster at evaluating `\if...'`s, switching to `universal` commands should take a minimum amount of time. 3) From a portability point of view, this way of shifting meanings of commands such as `\textquotedblleft`, according to context, doesn't screw up older documents where the `uni` package is added to.

There are some disadvantages. 1) Most notably there is *no* conformation to *any* existing encoding, which however is a problem of the font, not the `uni` package. 2) The heavy use of `\def`, `\edef`, and `\let`, is not exactly $\LaTeX 3$ policy, but is needed here. 3) The definition of the commands may slow \TeX down a bit, but should be bearable.

```

17 \newcommand{\uni@init}{\@unittrue}
18 \DeclareRobustCommand{\Declare@Uni}[1]{%
19   \edef\temp@{\expandafter@gobble\string#1}
20   \edef\temp@a{\csname uni@\temp@\endcsname}
21   \ifundefined{\temp@}{%
22     \edef\temp@c{%
23       \noexpand\if@uni\noexpand\aftergroup\temp@a\noexpand%
24       \else\noexpand\UniError{#1}\noexpand\fi}}{%
25     \edef\temp@b{\csname no@uni@\temp@\endcsname}
26     \edef\temp@c{%
27       \noexpand\if@uni\noexpand\aftergroup\temp@a\noexpand%
28       \else\noexpand\aftergroup\temp@b\noexpand\fi}}
29   \expandafter\let\temp@b#1}}
30 \DeclareRobustCommand{\DeclareUniChar}[2]{%
31   \Declare@Uni{#1}
32   \expandafter\chardef\temp@a=#2
33   \let#1\temp@c}

```

```

34 \def\DeclareUniCommand#1{%
35   \@ifnextchar[{\DeclareUni@xarg#1}{\DeclareUni@narg#1}}
36 \def\DeclareUni@xarg#1[#2]#3{%
37   \Declare@Uni{#1}
38   \expandafter\DeclareRobustCommand\temp@a[#2]{#3}
39   \let#1\temp@c}
40 \def\DeclareUni@narg#1#2{%
41   \Declare@Uni{#1}
42   \expandafter\DeclareRobustCommand\temp@a{#2}
43   \let#1\temp@c}
44 \@onlypreamble\DeclareUniChar\relax
45 \@onlypreamble\DeclareUniCommand\relax
46 \DeclareRobustCommand{\UniError}[1]{%
47   \PackageError{uni}{%
48     Command \string#1 not defined outside universal font.}{%
49     Correct your source file, so that \string#1 doesn't appear
50     ouside of universal font, then run LaTeX again.^^J%
51     Prepare your self for some gastly output.}}
52 % \expandafter\show\csname no@uni@\temp@\endcsname
53 % \expandafter\show\csname uni@\temp@\endcsname
54 % \expandafter\show\csname \temp@\endcsname
55

```

• 15

<pre> \bausquare \baucircle \bautriangle \bauhead \bauforms \dh \ng \th \varQ \NG \TH \textogonek \textbraceleft \textbraceright \textsection \textbar \guilsinglleft \guilsinglright \textquotedblleft \textquotedblright \quotesinglbase \quotedblbase \guillemotleft \guillemotright \k \DH \DJ \dj \textunderscore </pre>	<p>This command takes care of commands for non-standard placement, and symbols in the <code>universal</code> font, while using standard L^AT_EX 2_ε encodings, such as OT1.</p> <p>For instance, the symbols <code>}</code>, <code>{</code> isn't present in Computer Modern fonts, but is accessible through the command <code>\{</code>, and <code>\}</code> in the <code>universal</code> font. On the other hand <code>Γ</code> isn't present in this font, but is replaced by ■.</p> <p>Some of the commands available are well-known to the T1 encoding, and are provided with the same command names as in that encoding.</p> <p>The declaration of these commands use my (slick) macros <code>\DeclareUniChar</code> and <code>\DeclareUniCommand</code> — very neat I think.</p> <pre> 56 \DeclareUniChar{\bausquare}{"00} 57 \DeclareUniChar{\baucircle}{"01} 58 \DeclareUniChar{\bautriangle}{"02} 59 \DeclareUniChar{\bauhead}{"03} 60 \DeclareUniChar{\bauforms}{"04} 61 \DeclareUniChar{\dh}{"05} 62 \DeclareUniChar{\ng}{"06} 63 \DeclareUniChar{\th}{"07} 64 \DeclareUniChar{\varQ}{"08} 65 \DeclareUniChar{\NG}{"09} 66 \DeclareUniChar{\TH}{"0A} 67 \DeclareUniChar{\textogonek}{"0B} 68 \DeclareUniChar{\textbraceleft}{"0C} 69 \DeclareUniChar{\textbraceright}{"0D} 70 \DeclareUniChar{\textsection}{"0E} 71 \DeclareUniChar{\textbar}{"0F} 72 \DeclareUniChar{\guilsinglleft}{"22} 73 \DeclareUniChar{\guilsinglright}{"5C} </pre> <p>¹⁵Change of <code>\bausquare</code> on 98/08/01, Version 2.0</p>
---	--


```

74 \DeclareUniChar{\quotesinglbase}{"2C}
75 \DeclareUniCommand{\quotedblbase}{\char"2C\kern-.3em\char"2C}
76 \DeclareUniCommand{\textquotedblleft}{‘}
77 \DeclareUniCommand{\textquotedblright}{’}
78 \DeclareUniCommand{\guillemotleft}{\char"22\char"22}
79 \DeclareUniCommand{\guillemotright}{\char"5C\char"5C}
80 \DeclareUniCommand{\DH}{\raise.5ex\hbox{\char"2D}\kern-.5em D}
81 \DeclareUniCommand{\DJ}{\raise.5ex\hbox{\char"2D}\kern-.5em D}
82 \DeclareUniCommand{\dj}{\raise.75ex\hbox{\char"2D}\kern-1em d}
83 \DeclareUniCommand{\textunderscore}{\lower.5ex\hbox{\char"7B}}
84 \DeclareUniCommand{\k}[1]{%
85     \leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent11 \#1%
86     \else\oalign{\unhbox\z@\cr\hidewidth\char11\hidewidth}\fi}
87

```

4.1.4 Font Selection

•¹⁶

`\unifamily` The real command. This switches to `universal` font, and can be used by authors in conjunction with L^AT_EX 2_ε primitive `\selectfont`. `\unifamily` is used below in `\textuni...`, and `\uni...`.

`\unifamilydefault` Here `\unifamilydefault` is by default defined to be `uni`, but could ofcourse expand to something different. However, the expansion should be a font family (like `cmr`) or the like.

```

88 \DeclareRobustCommand\unifamily{%
89     \not@math@alphabet\unifamily\relax%
90     \fontfamily\unifamilydefault\selectfont}
91 \newcommand{\unifamilydefault}{uni}
92

```

•¹⁷

`\uniserries` If the `bold` option is used with the `uni` package, then this command switches to whatever series is defined via `\uniserriesdefault` (default is `bold` series), otherwise it does nothing. Since this command is used in `\textuni` and `\uni`, this way will not allow anyother series if `bold` option is given, expect `bold`. However, if `bold` isn't given, then *any* series is possible.

`\uniserriesdefault` Here `\uniserriesdefault` is defined to be `b (medium)` if the `bold` option *was* given to `uni` package. Otherwise it expands to nothing (`\relax`). This macro is used by `\uniserries`.

```

93 \ifm@dium
94     \let\uniserries\relax
95     \let\uniserriesdefault\relax
96 \else
97     \DeclareRobustCommand\uniserries{%
98         \not@math@alphabet\uniserries\relax%
99         \fontseries\uniserriesdefault\selectfont}
100    \newcommand{\uniserriesdefault}{b}
101 \fi
102

```

¹⁶Change of `\unifamily` on 98/08/01, Version 2.0

¹⁷Change of `\uniserries` on 98/08/01, Version 2.0

•18

`\unishape` If the `strict` option is used with the `uni` package, then this command switches to whatever shape is defined via `\unishapedefault` (default is *strict* shape), otherwise it does nothing. Since this command is used in `\textuni` and `\uni`, this way will not allow anyother shape if `strict` option is given, expect *strict*. However, if `strict` isn't given, then *any* series is possible.

`\unishapedefault` Here `\unishapedefault` is defined to be `st` (*strict*) if the `strict` option *was* given to `uni` package. Otherwise it expands to nothing (`\relax`). This macro is used by `\unishape`.

```
103 \ifstri@t
104   \DeclareRobustCommand\unishape{%
105     \not@math@alphabet\unishape\relax%
106     \fontseries\unishapedefault\selectfont}
107   \newcommand{\unishapedefault}{st}
108 \else
109   \let\unishape\relax
110   \let\unishapedefault\relax
111 \fi
```

`\stshape` This command selects the shape holded by `\stdefault` (defaults to `st` — *strict*) in any family or series. This is used by commands `\textunist`, `\textunibst`, `\unist`, and `\unibst`.

`\stdefault` The shape selected by `\stshape`.

```
112 \DeclareRobustCommand\stshape{%
113   \not@math@alphabet\stshape\relax%
114   \fontshape\stdefault\selectfont}
115 \newcommand{\stdefault}{st}
116
```

Local Font Selection

•19

`\textuni` These are the macros that switches to `universal` font, locally, i.e., for a few words, `\textunirm` in various series and shapes. The names are pretty selfexplanatory, I think. These `\textunibf` macros will *always* change the font series and shape accroding to it's definiton, `\textunisl` nothing more, and nothing less, exepct `\textuni`, which only changes the cuurent `\textunisc` family to `uni`. Notice it simply uses the `\unifamily` command to do the trick.

`\textunibsl` The second argument to `\DeclareTextFontCommand` should not normally contain commands that typeset, or commands not relevant to the selection of fonts, `\textunibsc` but to make up for the non–default characters and character placements, we *do* include `\uni@init` in the argument.

Notice that only `\textuni` is defined if option `strict` was given to `uni` package.

```
117 \DeclareTextFontCommand{\textuni}{%
118   \uni@init\unifamily\uniseries\unishape}
119 \ifstri@t
120 \else
121   \DeclareTextFontCommand{\textunirm}{%
```

¹⁸Change of `\unishape` on 98/08/01, Version 2.0

¹⁹Change of `\textuni` on 98/08/01, Version 2.0

```

122   \uni@init\unifamily\mdseries\upshape}
123   \DeclareTextFontCommand{\textunibf}{%
124   \uni@init\unifamily\bfseries\upshape}
125   \DeclareTextFontCommand{\textunisl}{%
126   \uni@init\unifamily\mdseries\slshape}
127   \DeclareTextFontCommand{\textunisc}{%
128   \uni@init\unifamily\mdseries\scshape}
129   \DeclareTextFontCommand{\textunist}{%
130   \uni@init\unifamily\mdseries\stshape}
131   \DeclareTextFontCommand{\textunibsl}{%
132   \uni@init\unifamily\bfseries\slshape}
133   \DeclareTextFontCommand{\textunibsc}{%
134   \uni@init\unifamily\bfseries\scshape}
135   \DeclareTextFontCommand{\textunibst}{%
136   \uni@init\unifamily\bfseries\stshape}
137 \fi
138

```

Global Font Selection

•²⁰

```

\uni   These macros works as \textuni... above, except the take effect from the point
\unibf used, to the end of the current group. Again font selection is exactly as given in
\unibf the command name, except for \uni which only changes the family.
\unisl   Notice that only \uni is defined if option strict was given to uni package.
\unisc 139 \DeclareOldFontCommand{\uni}{%
\unibsl 140 \uni@init\unifamily\uniserie\unishape}{}
\unibsc 141 \ifstri@t
142 \else
143   \DeclareOldFontCommand{\unirm}{%
144   \uni@init\unifamily\mdseries\upshape}{}
145   \DeclareOldFontCommand{\unibf}{%
146   \uni@init\unifamily\bfseries\upshape}{}
147   \DeclareOldFontCommand{\unisl}{%
148   \uni@init\unifamily\mdseries\slshape}{}
149   \DeclareOldFontCommand{\unisc}{%
150   \uni@init\unifamily\mdseries\scshape}{}
151   \DeclareOldFontCommand{\unist}{%
152   \uni@init\unifamily\mdseries\stshape}{}
153   \DeclareOldFontCommand{\unibsl}{%
154   \uni@init\unifamily\bfseries\slshape}{}
155   \DeclareOldFontCommand{\unibsc}{%
156   \uni@init\unifamily\bfseries\scshape}{}
157   \DeclareOldFontCommand{\unibst}{%
158   \uni@init\unifamily\bfseries\stshape}{}
159 \fi
160

```

default Option Font Now if you gave the `default` option to the package, `\ifdef@ult` evaluates to true, and so we setup the default font to be `universal`,

²⁰Change of \uni on 98/08/01, Version 2.0

in the medium or bold version, depending on whether you gave the `medium` option or not.

```

161 \ifdef@ult
162   \renewcommand{\familydefault}{\uni}
163   \ifm@dium\renewcommand{\seriesdefault}{m}
164   \else\renewcommand{\seriesdefault}{b}\fi
165   \renewcommand{\itdefault}{sl}

\cmrfamily We define \cmrfamily and \cmrdefault to change back to Computer Modern Ro-
\cmrdefault man font, if so wanted. The macros \textcmr and \cmr are the logical extensions
\textcmr of \cmrfamily.
\cmr 166 \DeclareRobustCommand\cmrfamily{%
167   \not@math@alphabet\cmrfamily\relax%
168   \fontencoding\cmrenc\fontfamily\cmrdefault\selectfont}
169 \newcommand{\cmrdefault}{\cmr}
170 \newcommand{\cmrenc}{OT1}
171 \DeclareTextFontCommand{\textcmr}{\cmrfamily}
172 \DeclareOldFontCommand{\cmr}{\cmrfamily}{}
173 \fi
174

```

4.2 The Font Definition Files — `*uni.fd`

These files are needed in the New Font Selection Scheme, used by $\text{\LaTeX} 2_{\epsilon}$, but really isn't necessary for pure \TeX or $\text{\LaTeX} 2.09$ users, but since those are treathned races, I did put in the extra effort and made the files.

What they really do, is to specify what font driver file should be loaded when the user wishes font. Notice, a lot of the font shapes etc. available in the Computer Modern Roman scheme isn't available in the `universal` font, so we substitute with whatever is closest, something from the `universal` or Computer Modern Roman scheme.

The reason why the file names may look a bit weird, is to make the font conform to the standard set by `fontname` used by `kpathsea` in most \TeX , \LaTeX , and `METAFONT` systems.

Math Encoding First comes the the font definition file for *math*, but since no math characters is defined in the `universal` font, we substitute with the relevant fonts from Computer Modern Roman.

```

1 \ProvidesFile{oamluni.fd}
2   [1998/08/01 v2.0 Non Standard LaTeX font definitions]
3 \DeclareFontFamily{OML}{uni}{\skewchar\font127 }
4 \DeclareFontShape{OML}{uni}{m}{n}{<-> ssub * cmm/m/it}{}
5 \DeclareFontShape{OML}{uni}{m}{it}{<-> ssub * cmm/m/it}{}
6 \DeclareFontShape{OML}{uni}{m}{sl}{<-> ssub * cmm/m/it}{}
7 \DeclareFontShape{OML}{uni}{m}{sc}{<-> ssub * cmm/m/it}{}
8 \DeclareFontShape{OML}{uni}{bx}{n}{<-> ssub * cmm/b/it}{}
9 \DeclareFontShape{OML}{uni}{bx}{it}{<-> ssub * cmm/b/it}{}
10 \DeclareFontShape{OML}{uni}{bx}{sl}{<-> ssub * cmm/b/it}{}
11 \DeclareFontShape{OML}{uni}{bx}{sc}{<-> ssub * cmm/b/it}{}
12

```

Symbols Encoding Next is the definitions for *symbols* fonts, but as above, there is no separate symbol font defined for the `universal` font, so we substitute for default.

```

1 \ProvidesFile{omsuni.fd}
2     [1998/08/01 v2.0 Non Standard LaTeX font definitions]
3 \DeclareFontFamily{OMS}{uni}{\skewchar\font48 }
4 \DeclareFontShape{OMS}{uni}{m}{n}{<-> ssub * cmsy/m/n}{-}{}
5 \DeclareFontShape{OMS}{uni}{m}{it}{<-> ssub * cmsy/m/n}{-}{}
6 \DeclareFontShape{OMS}{uni}{m}{sl}{<-> ssub * cmsy/m/n}{-}{}
7 \DeclareFontShape{OMS}{uni}{m}{sc}{<-> ssub * cmsy/m/n}{-}{}
8 \DeclareFontShape{OMS}{uni}{bx}{n}{<-> ssub * cmsy/b/n}{-}{}
9 \DeclareFontShape{OMS}{uni}{bx}{it}{<-> ssub * cmsy/b/n}{-}{}
10 \DeclareFontShape{OMS}{uni}{bx}{sl}{<-> ssub * cmsy/b/n}{-}{}
11 \DeclareFontShape{OMS}{uni}{bx}{sc}{<-> ssub * cmsy/b/n}{-}{}
12

```

Normal Encoding Now the definitions for the normal font. This *is* of course defined, anything else would be ludicrous. It is very straight forward most of the way. However, notice the substitutions at the end of the file.

```

1 \ProvidesFile{ot1uni.fd}%
2     [1998/08/01 v2.0 Non standard LaTeX font definitions]
3 \DeclareFontFamily{OT1}{uni}{\hyphenchar\font45 }
4

```

Next comes the specifications of what to load in normal upright shape

```

5 \DeclareFontShape{OT1}{uni}{m}{n}{
6     <5><6><7><8>fulmr8
7     <9>fulmr9
8     <10><10.95>fulmr10
9     <12><14.4>fulmr12
10    <17.28><20.74><24.88>fulmr17
11 }{}
12

```

Next comes the specifications of what to load in *normal slanted shape*

```

13 \DeclareFontShape{OT1}{uni}{m}{sl}{
14     <5><6><7><8>fulmo8
15     <9>fulmo9
16     <10><10.95>fulmo10
17     <12><14.4>fulmo12
18     <17.28><20.74><24.88>fulmo17
19 }{}
20

```

Now for `SMALL CAPS MEDIUM` definitions.

```

21 \DeclareFontShape{OT1}{uni}{m}{sc}{
22     <5><6><7><8>fulmc8
23     <9>fulmc9
24     <10><10.95>fulmc10
25     <12><14.4>fulmc12
26     <17.28><20.74><24.88>fulmc17
27 }{}
28

```

Now for strict medium definitions.

```
29 \DeclareFontShape{OT1}{uni}{m}{st}{
30 <5><6><7><8>fulmst8
31 <9>fulmst9
32 <10><10.95>fulmst10
33 <12><14.4>fulmst12
34 <17.28><20.74><24.88>fulmst17
35 }{}
36
```

Next comes the specifications of what to load in bold face upright shape

```
37 \DeclareFontShape{OT1}{uni}{b}{n}{
38 <5><6><7><8>fulbr8
39 <9>fulbr9
40 <10><10.95>fulbr10
41 <12><14.4>fulbr12
42 <17.28><20.74><24.88>fulbr17
43 }{}
44
```

Next comes the specifications of what to load in bold face slanted shape

```
45 \DeclareFontShape{OT1}{uni}{b}{sl}{
46 <5><6><7><8>fulbo8
47 <9>fulbo9
48 <10><10.95>fulbo10
49 <12><14.4>fulbo12
50 <17.28><20.74><24.88>fulbo17
51 }{}
52
```

And small caps bold face definitions.

```
53 \DeclareFontShape{OT1}{uni}{b}{sc}{
54 <5><6><7><8>fulbc8
55 <9>fulbc9
56 <10><10.95>fulbc10
57 <12><14.4>fulbc12
58 <17.28><20.74><24.88>fulbc17
59 }{}
60
```

And strict bold face definitions.

```
61 \DeclareFontShape{OT1}{uni}{b}{st}{
62 <5><6><7><8>fulbst8
63 <9>fulbst9
64 <10><10.95>fulbst10
65 <12><14.4>fulbst12
66 <17.28><20.74><24.88>fulbst17
67 }{}
68
```

Now for the substitutions. This is straight forward, that is, upright slanted is substituted by upright; bold face condensed, and bold face extra is substituted by boldface; and italic by slanted²¹.

²¹As mentioned earlier, one of the characteristics of this dont, is that it has no serifs, so it would be strange to include italics in this font.

```

69 \DeclareFontShape{OT1}{uni}{m}{it}{<->ssub*uni/m/sl}{}
70 \DeclareFontShape{OT1}{uni}{m}{ui}{<->ssub*uni/m/n}{}
71 \DeclareFontShape{OT1}{uni}{b}{it}{<->ssub*uni/b/sl}{}
72 \DeclareFontShape{OT1}{uni}{bx}{n}{<->ssub*uni/b/n}{}
73 \DeclareFontShape{OT1}{uni}{bx}{sl}{<->ssub*uni/b/sl}{}
74 \DeclareFontShape{OT1}{uni}{bx}{sc}{<->ssub*uni/b/sc}{}
75 \DeclareFontShape{OT1}{uni}{bx}{st}{<->ssub*uni/b/st}{}
76 \DeclareFontShape{OT1}{uni}{bx}{it}{<->ssub*uni/b/sl}{}
77 \DeclareFontShape{OT1}{uni}{bc}{ui}{<->ssub*uni/b/n}{}
78 \DeclareFontShape{OT1}{uni}{sbc}{n}{<->ssub*uni/b/n}{}
79

```

T1 encoding Since the T1 encoding relies on the font scheme of `exbase` (EC fonts), and `dxbase` (DC fonts), by Jörg Knappen and Nibert Schwartz, which is a whole new way of making fonts, that requires some special measures that I couldn't incorporate into the `universal` font without a whole lot of work, I chose not to make support for the T1 scheme. Therefore this file issues a warning and then exists the whole `LATEX` run. This may seem a bit drastic, but it ensures that you don't get screwed up results.

```

1 \ProvidesFile{t1uni.fd}
2     [1998/08/01 v2.0 Non Standard LaTeX font definitions]
3 \typeout{WARNING: universal font not available in t1 definition.}
4 \typeout{WARNING: exiting, correct your source.}
5 \stop
6

```

U Encoding This is similar to the normal encoding above, except encoding is U.

```

1 \ProvidesFile{uuni.fd}
2     [1998/08/01 v2.0 Non Standard LaTeX font definitions]
3 \DeclareFontFamily{U}{uni}{\hyphenchar\font45}
4 \DeclareFontShape{U}{uni}{m}{n}{
5     <5><6><7><8>fulmr8
6     <9>fulmr9
7     <10><10.95>fulmr10
8     <12><14.4>fulmr12
9     <17.28><20.74><24.88>fulmr17
10    }{}
11 \DeclareFontShape{U}{uni}{m}{sl}{
12     <5><6><7><8>fulmo8
13     <9>fulmo9
14     <10><10.95>fulmo10
15     <12><14.4>fulmo12
16     <17.28><20.74><24.88>fulmo17
17    }{}
18 \DeclareFontShape{U}{uni}{m}{sc}{
19     <5><6><7><8>fulmc8
20     <9>fulmc9
21     <10><10.95>fulmc10
22     <12><14.4>fulmc12
23     <17.28><20.74><24.88>fulmc17
24    }{}

```

```


25 \DeclareFontShape{U}{uni}{m}{st}{
26   <5><6><7><8>fulmst8
27   <9>fulmst9
28   <10><10.95>fulmst10
29   <12><14.4>fulmst12
30   <17.28><20.74><24.88>fulmst17
31 }{}
32 \DeclareFontShape{U}{uni}{b}{n}{
33   <5><6><7><8>fulbr8
34   <9>fulbr9
35   <10><10.95>fulbr10
36   <12><14.4>fulbr12
37   <17.28><20.74><24.88>fulbr17
38 }{}
39 \DeclareFontShape{U}{uni}{b}{sl}{
40   <5><6><7><8>fulbo8
41   <9>fulbo9
42   <10><10.95>fulbo10
43   <12><14.4>fulbo12
44   <17.28><20.74><24.88>fulbo17
45 }{}
46 \DeclareFontShape{U}{uni}{b}{sc}{
47   <5><6><7><8>fulbc8
48   <9>fulbc9
49   <10><10.95>fulbc10
50   <12><14.4>fulbc12
51   <17.28><20.74><24.88>fulbc17
52 }{}
53 \DeclareFontShape{U}{uni}{b}{st}{
54   <5><6><7><8>fulbst8
55   <9>fulbst9
56   <10><10.95>fulbst10
57   <12><14.4>fulbst12
58   <17.28><20.74><24.88>fulbst17
59 }{}
60 \DeclareFontShape{U}{uni}{m}{it}{<->ssub*uni/m/sl}{}
61 \DeclareFontShape{U}{uni}{m}{ui}{<->ssub*uni/m/n}{}
62 \DeclareFontShape{U}{uni}{b}{it}{<->ssub*uni/b/sl}{}
63 \DeclareFontShape{U}{uni}{bx}{n}{<->ssub*uni/b/n}{}
64 \DeclareFontShape{U}{uni}{bx}{sl}{<->ssub*uni/b/sl}{}
65 \DeclareFontShape{U}{uni}{bx}{sc}{<->ssub*uni/b/sc}{}
66 \DeclareFontShape{U}{uni}{bx}{st}{<->ssub*uni/b/st}{}
67 \DeclareFontShape{U}{uni}{bx}{it}{<->ssub*uni/b/sl}{}
68 \DeclareFontShape{U}{uni}{bc}{ui}{<->ssub*uni/b/n}{}
69 \DeclareFontShape{U}{uni}{sbc}{n}{<->ssub*uni/b/n}{}
70

```


A Solution to the \bauhausforms problem

A.1 The Problem

This I owe to Joseph Collins <collins@ARL.MIL>.

During the preparation of version 1.0 of this font, I was loosing sleep over what I chose to call ‘The \bauhausforms problem’, after the problems the symbol  gave me.

As can be seen from the symbol, the idea is to make a figure out of a circle, a square, and a triangle. From these three figures you can ofcourse make infinitely many figures, even though the sizes are limited. The particular combination of the three figures I was looking, first seemed simple²⁹, but turned out to be difficult. What I wanted to do was (see also figure 1):

Take a circle of some radius (e.g., 1). Inside this circle draw a isosceles triangle $\triangle ABC$, with all vertices on the circle, The singular vertex A placed on the horizontal line traveling left from the center of the circle.

Now draw a square $\square PQRS$ inside of the circle, having two vertices on the circle, and two on the triangle.

The wanted triangle and square are such that, the opposing side of the singular vertex BC should divide the square into two equal oblongs, i.e., intersect PQ midway between P and Q .

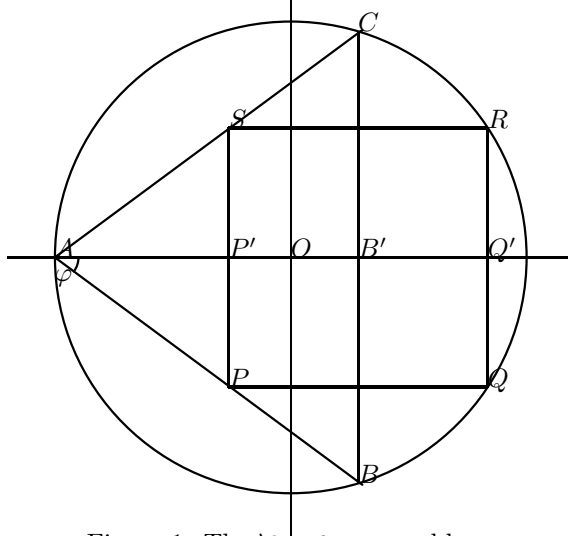


Figure 1: The \bauforms problem.

This shouldn't be too difficult, should it. Well I didn't think so, but after many late evenings with pen, paper, ruler, compasses, and heavy use of trigonometric relations, I found out that the half φ of the singular vertex should obey:

$$0 = \sin \left(\cos^{-1} \left(\frac{\cos 2\varphi}{\sqrt{2}} \right) \right) - \frac{\cos 2\varphi}{\sqrt{2}} - 2\sqrt{2} \frac{\cos \varphi^2 \sin \varphi}{\cos \varphi + \sin \varphi}$$

²⁹And after having seen Mr. Collins solution, it did again.

Now I dare you to find the exact solution to that.

Using numerical methodes (Newton's method), was ofcourse no problem, and gave satesfatory resualt. If the expression on the right above is labelled f , f' is:

$$f' = \frac{\sin 4\varphi}{2\sqrt{\frac{3-\cos 4\varphi}{4}}} + \sqrt{2} \sin 2\varphi - \frac{2\sqrt{2} \cos \varphi}{\cos \varphi + \sin \varphi^2} ((\cos \varphi + \sin \varphi) (\cos^2 \varphi - 2 \sin^2 2\varphi) - (\cos \varphi - \sin \varphi) \cos^2 \varphi \sin \varphi)$$

Using these expressions for f and f' in a Fortran program, I reached resaults close to what Mr. Collins found.

A.2 Joseph Collins' Solution

In Mr. Collins notation, the points on figure 1 has the following coordinates:

$$\begin{aligned} A &= (0, -1) & B' &= (x, 0) & C &= (x, y) \\ O &= (0, 0) & P' &= (x - h, 0) & Q' &= (x + h, 0) \\ R &= (x + h, h) & S &= (x - h, h) \end{aligned}$$

Below is what Mr. Collins wrote me — thank you very much.

On the unit circle

$$x^2 + y^2 = 1 \tag{1}$$

we have the vertices of a triangle at $(-1, 0)$, (x, y) , and $(x, -y)$. A square has four vertices $(x \pm h, \pm h)$, where the two points $(x - h, \pm h)$ lie on the triangle (constraint A) and the two points $(x + h, \pm h)$ lie on the circle (constraint B). Thus, the vertical side of the triangle bisects the square. From constraint A, upon consideration of similar triangles, we have

$$\frac{y}{1+x} = \frac{h}{1+x-h}, \quad \text{so that} \quad h = \frac{(1+x)y}{1+x+y}.$$

By (1), this is

$$h = \frac{(1+x)\sqrt{1-x^2}}{1+x+\sqrt{1-x^2}}. \tag{2}$$

From constraint B, we get

$$(x+h)^2 + h^2 = 1. \tag{3}$$

Any simultaneous solution of (2) and (3) is also a solution of

$$8x^3 - 4x^2 - 3x + 1 = 0, \tag{4}$$

the relevant solution being

$$x = \frac{1}{6} + \sqrt{\frac{11}{18}} \sin \left[\frac{\pi}{6} - \frac{1}{3} \arctan \left(\frac{3\sqrt{237}}{23} \right) \right]. \tag{5}$$

Equation (4) and solution (5) courtesy of Mathematica. We get y and h from (1) and (2), respectively. The angle at $(-1, 0)$ is $\varphi = 2 \arctan(y/(1+x))$.

Approximate values are

$$\begin{aligned}x &\simeq 0.2865914 \\y &\simeq 0.9580529 \\h &\simeq 0.5491394 \\\varphi &\simeq 1.280129 (\simeq 73.346^\circ)\end{aligned}$$

B Copyrights — GNU General Public License

This METAFONT implementation of the “universal” font and the L^AT_EX support package “uni” copyright © 1998 Christian Holm.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to

The Free Software Foundation, Inc.
675 Mass Ave
Cambridge
MA 02139
USA

See the file `copyright` in the distribution for the complete GNU General Public License.

You can reach me (the copyright holder) at

Christian Holm
Sankt Hansgade 23, 1. th.
DK-2200 Copenhagen N
Denmark
E-mail: `cholm@nbi.dk` or `cholm@fys.ku.dk`

C Wishlist

Below is a list of things I would like to do with the font and package. If anyone has any suggestions, ready-made code, or new ideas, please let me know.

If you would like to take on one or more of the tasks presented below, please do so, but send me a note so that I may coordinate with my own efforts, and perhaps have a constructive discourse.

I should however instruct you to read the *complete* documentation of the package and font, since this may give some reasons why I have chosen a particular approach.

- Make the font an 8-bit encoded (256 characters) font, conforming somewhat to the T1 encoding of the Cork fonts.
- Making a package (perhaps `mfdoc`), to make documentation of METAFONT sources, just like `doc` is for \LaTeX sources. This is ofcourse a independent project, and I will proberly not work on it.

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	■	●	▲	⊙	⊕	ð	ŋ	þ	"0x
'01x	ɑ	Ɔ	Ɔ	.	{	}	§		
'02x			"1x
'03x	.	β	œ	œ	ø	Æ	Œ	Ø	
'04x	-	!	<	#	\$	%	&	'	"2x
'05x	()	*	+	.	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	:		=	¿	?	
'10x	Q	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[>]	.	.	
'14x	.	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	-	-	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

D Font Charts

Below are some charts of the **universal** font in different series and shapes (medium upright, bold upright, medium slanted, medium small caps), all in size 8pt.

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	■	●	▲	⌚	⊕	∂	η	β	"0x
'01x	q	0	p	,	{	}	§		
'02x			"1x
'03x	,	ß	œ	œ	ø	Æ	€	∅	
'04x	-	!	<	#	\$	%	8	'	"2x
'05x	()	*	+	.	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	:		=	¿	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[>]	.	.	
'14x	.	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	-	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	■	●	▲	⌚	⊕	q	p	p	"0x
'01x	q	0	p	,	{	}	§		
'02x			"1x
'03x	,	ss	Æ	€	ø	Æ	€	∅	
'04x	-	!	<	#	\$	%	8	'	"2x
'05x	()	*	+	.	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	:		=	¿	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[>]	.	.	
'14x	.	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	-	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	■	●	▲	⊙	⊕	∂	η	þ	"0x
'01x	∂	η	þ	.	{	}	§		
'02x			"1x
'03x	.	β	œ	œ	ø	œ	œ	ø	
'04x	-	!	<	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	∅	4	5	6	7	"3x
'07x	8	9	:	:		=	¿	?	
'10x	Q	a	b	c	d	e	f	q	"4x
'11x	h	i	j	k	l	m	n	o	
'12x	p	q	r	s	t	u	v	w	"5x
'13x	x	y	z	[>]	.	.	
'14x	`	a	b	c	d	e	f	q	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	-	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	■	●	▲	⊙	⊕	∂	η	þ	"0x
'01x	q	o	p	.	{	}	§		
'02x			.	.	v	v	-	.	"1x
'03x	.	β	œ	œ	ø	Æ	€	Ø	
'04x	-	!	<	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	∅	4	5	6	7	"3x
'07x	8	9	:	;		=	¿	?	
'10x	Q	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[>]	^	.	
'14x	`	a	b	c	d	e	f	q	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	-	-	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	