Fonts

Implementing the extended TEX layout using PostScript fonts

Sebastian Rahtz

1 Introduction

When Donald Knuth made virtual fonts part of the general TEX repertoire at the end of 1989 [5], at the same time as extending both TEX and METAFONT to support 8-bit input, it immediately became obvious that the TEX world needed a new standard for the layout of fonts. This is not the place to review the discussion of what characters should be in a generalized 256 character text font (see [4, 3]), nor to pass judgement on the proposed solution, finalized at the TEX Users Group meeting in Cork in September 1990, but rather to demonstrate how the layout can be implemented using POSTSCRIPT fonts (Donald Knuth and Tom Rokicki have already shown how to create POSTSCRIPT fonts in the original TEX layout, using virtual fonts, implemented in the latter's distributed `afm2tfm` program). To remind ourselves what we are aiming at, Fig. 1 shows the new layout using the Extended Computer Modern Roman font (`dcr10`, from the `dc` family created by Norbert Schwarz). Some characters needed extra work with METAFONT, but most were created from existing building blocks, and we shall follow this approach with the POSTSCRIPT fonts.

The set of procedures described below has only been made to work in a limited environment, but the principles can be used for most common `dvi`-to-POSTSCRIPT drivers, so long as they support virtual fonts. Our assumption in what follows is that we are using Rokicki's `afm2tfm` program (to convert Adobe font metrics to TEX font metrics) and the same author's `dvips` program, currently the only public domain `dvi` to POSTSCRIPT program to implement virtual fonts. The system has only been tested under Unix and OS/2, but should be easily portable, consisting of a program written in C, and the standard TEXware programs.

Examples are given in a Lucida Bright font.

2 Strategy

It may be necessary to remind readers that normal POSTSCRIPT fonts are essentially arranged as a set of procedures which use a lookup table of *names* for the characters; generally, fonts include a mapping between the names and numbers, but this is easily changed. Thus the character `exclamdown` (¡) is *normally* mapped to decimal 161, but it can always be called by name, or mapped to another number. The standard Adobe mapping is given in [1], and corresponds to no obvious standard; the layout is as shown in Fig. 2.

We can use straightforward methods to do the basic redefinition of characters so that they suit the extended layout. Firstly, we follow Rokicki's lead in `afm2tfm` and build up a virtual font which fools TEX into thinking that character X is at position A in the font, when it is in fact at position B; since we shall need virtual fonts anyway, this is an economic solution. Until version 7.0 of `afm2tfm`, this was made difficult by the fact that `afm2tfm` mapped undefined characters in an Adobe font more or less randomly; these are characters not assigned a font position in the Adobe Font Metric file (such as composite letter/accent combinations) which we want to use at specific positions. This required changing the AFM file (using a simple program to automate the process) so that characters were given the number we actually wanted—this forced `afm2tfm` to allocate them correctly. Thus the AFM entry

```
C -1 ; WX 444 ; N zcaron ; B 25 0 418 674 ;
```

was changed to

```
C 186 ; WX 444 ; N zcaron ; B 25 0 418 674 ;
```

This also needed a slightly revised version of the `afm2tfm` program itself, as it used to have a fixed table listing the first 128 characters in the font (in traditional TEX layout). I implemented this change by having `afm2tfm` read an encoding table from an external file.

With version 7.0 onwards of `afm2tfm`, Rokicki provided a more general mechanism for changing the encoding of a font at successive stages (at the level of the virtual font, and also at the level of the PostScript font itself). This has made the process described here rather easier.

When we have constructed a suitable AFM file, we can run `afm2tfm` with the option to generate a `.vpl` file; but this only rearranges the existing characters—what about the symbols which are not in the original font? These fall into four groups:

- New composite letter/accent combinations like 'S acute' (**Ś**) which can be generated using the available floating accents.
- Composite characters which can be 'fudged' using existing letters, such as the pseudo-character for capital **ß** (**SS**) or 'ij' (**ij**).
- New glyphs, such as the visible space character (which can be created using rules, as in this experiment) or 'dotless j' (**J**).
- New ligatures not provided in POSTSCRIPT fonts, such as ffi (**ffi**).

	'0	*'1*	*'2*	*'3*	*'4*	*'5*	*'6*	*'7*	
'00x	`	´	^	~	¨	˝	°	ˇ	"0x
'01x	˘	¯	˙	¸	˛	‚	‹	›	
'02x	“	”	„	«	»	–	—		"1x
'03x	ı	ȷ	ff	fi	fl	ffi	ffl		
'04x	␣	!	"	#	$	%	&	’	"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	{	|	}	~	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8x
'21x	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ	
'22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	"9x
'23x	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§	
'24x	ă	ą	ć	č	ď	ě	ę	ğ	"Ax
'25x	Í	ľ	ł	ń	ň	ŋ	ő	ŕ	
'26x	ř	ś	š	ş	ť	ţ	ű	ů	"Bx
'27x	ÿ	ź	ž	ż	ij	¡	¿	£	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 1: Extended TEX layout

	´0	´1	´2	´3	´4	´5	´6	´7	
´04x		!	"	#	$	%	&	'	"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	{	|	}	~		
´24x		¡	¢	£	/	¥	ƒ	§	"Ax
´25x	¤	'	"	«	‹	›	fi	fl	
´26x		–	†	‡	·		¶	•	"Bx
´27x	‚	„	"	»	…	‰		¿	
´30x		`	´	ˆ	¯	˘	˙	¨	"Cx
´31x	"		°	¸		˝	˛	ˇ	
´32x	—								"Dx
´33x									
´34x		Æ		ª					"Ex
´35x	Ł	Ø	Œ	º					
´36x		æ				ı			"Fx
´37x	ł	ø	œ	ß					
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 2: Unadulterated Adobe font layout (using AvantGarde)

```
(CHARACTER O 221 (comment Sacute)
   (CHARWD R 525.00)
   (CHARHT R 908.00)
   (CHARDP R 20.00)
   (MAP
      (SETCHAR C S)
      (MOVERIGHT R -429.00)
      (MOVEUP R 231.00)
      (SETCHAR O 1)
      )
   )
```

Figure 3: Virtual font file entry for S acute

```
(CHARACTER O 274 (comment ij)
   (CHARWD R 425.00)
   (CHARHT R 688.00)
   (CHARDP R 283.00)
   (MAP
      (SETCHAR C i)
      (MOVERIGHT R -100.00)
      (SETCHAR C j)
      )
   )
```

Figure 4: Virtual font file entry for ij

All of these need a character added to the virtual font; each virtual font character description consists of metrics for character width, height and depth (and italic correction where appropriate), and some instruction for what to do. These instructions usually involve setting one or more characters from base fonts, and inserting vertical or horizontal movement (which can, of course, be negative). Typical entries are shown in Figs. 3 and 4.

Making accent/letter combinations work involves setting a letter, moving back to the right, and then setting the accent. Assuming that accents go onto the centre of a letter, the movement to the right can be calculated automatically from the width of the letter and the accent. In some cases there must also be an upward movement to place an accent over, for instance, a capital letter. This upward movement would be a problem were it not for the fact that most fonts are consistently designed, and accents are already at the right height to sit over lower-case letters. The upward (or downward for sub-letter accents) movement can be observed in the AFM file which supplies model instructions for the creation of the normal composites; this example gives the right and upward movement of the acute over a capital E:

```
CC Eacute 2 ; PCC E 0 0 ; PCC acute 139 214 ;
```

Some cases will present special problems, such as 'l acute' (ĺ), but an algorithm can be developed for each of these characters which seems to be consistent across fonts; thus 'l acute' needs the accent raising more than normal. For those fonts which lack characters like Thorn (þ) and eth (ð), these have to be fudged in a way copied from the original Cork demonstration chart.

To create completely new characters, we may need to resort to coding directly in PostScript; for this demonstration, I experimented with creating an extra font which contained just four new characters. Luckily, the code to do this had already been derived, and posted in a Usenet news group, by Amanda Walker. Greater patience would probably permit these characters to be generated entirely by `\special` commands in the virtual font, which would avoid the need for a font metric file for the tiny font. The code for creating the new font is given in Appendix B.

3 Usage

The approach outlined in the previous section resulted in two programs written in C, `vpltovpl` and `afm2afm`, and a slightly amended `afm2tfm` program. The latter two were made redundant by `afm2tfm` 7.0. Once the program `vpltovpl` has been compiled, the procedure for generating the final font is as follows, with examples shown in Unix syntax for a font 'Times-Roman' with an AFM file called `ptmr.afm`; we generate a new font called `ptmrq` to distinguish it from 'normal' fonts, utilising a 'raw' font called `ptmr0`.[1]

1. Run `afm2tfm` on the AFM file to generate a `tfm` file for an unmapped version of the font, and a `vpl` file describing the virtual font. The mapping is taken from an encoding file `ec.enc`, and the `-T` option means that the encoding is to be used both for the virtual font layout, and also as instructions to the PostScript interpreter to change the internal mapping of the font to make the normally inaccessible characters visible. The start of `ec.enc` is listed in Fig. 6.

   ```
   afm2tfm ptmr.afm -T ec.enc -v ptmrq.vpl \
         pmtr0.tfm
   ```

2. Run `vpltovpl` on the `vpl` file, also telling it the name of the `afm` file so that it can work out accent corrections etc.

   ```
   vpltovpl ptmrq.vpl ptmr.afm
   ```

[1] This uses the 'standard' names proposed by Karl Berry, suffixing them with a variant letter indicating an encoding.

This adds virtual character entries to the end of the `.vpl` file

3. We can now run the normal `vptovf` program which creates the `tfm` file which TEX will actually use, and the `vf` file which drivers will access.

   ```
   vptovf ptmrq.vpl ptmrq.vf ptmrq.tfm
   ```

4. We have to tell `dvips` about the fonts we have created, so that it can resolve references to, e.g., `ptmrq` in the virtual font. This is done by lines in the control file `psfonts.map`, which also instructs `dvips` to send the (same) encoding file `ec.enc` to the POSTSCRIPT interpreter to set up the font correctly. Some fonts may also need downloading. Some examples lines are given in Figure 5.

   If we use the public domain `ps2pk` program (created by Piet Tutelaars using IBM's library for rendering POSTSCRIPT Type1 fonts), we could instead generate TEX `.pk` fonts using exactly the same encoding file.

The result is a `tfm` file for TEX (`ptmrq.tfm`), and a `vf` file for the driver (`ptmrq.vf`), which makes references to the 'raw' font `ptmr0`.

Before we can go ahead and use the font in TEX, there is one more job: rebuilding some of (LA)TEX's standard macros to do with accents and special characters which have hard-wired character positions hardwired. A sample set of code is given in Appendix A.

4 Using the result

The final product of the work is no more than a font table; Fig. 7 shows the result using Times Roman. The 'visible space' (which was absent in earlier versions on this system) has been created with `SETRULE` commands in the virtual font. In most of the modern Adobe fonts, the more awkward glyphs already exist, but some of the characters are rather badly fudged, as a comparison with the DC chart shows. A run with LucidaBright derived small caps (Fig. 8) shows that the procedure works with small caps as well as with normal roman.

5 Remaining problems

There are problems left for the TEX community to solve in their relationship with POSTSCRIPT, even if we agree on the suggested layout — and it must be said that there are many people unhappy with the proposal. Thus the 'visible space' character is an extremely specialized brute, which, in the opinion of the author, has no place in a normal text font, and should be banished to a symbol font.

1. There remain some characters which are inadequately defined:
   (a) The 'Eng' (**ŋ**) and 'eng' (**Ŋ**) characters need total reworking; their creation from 'r' and 'j' is ridiculous; the original Cork table used this temporarily to show what was intended, but the combination was not designed to be useable.
   (b) The 'd bar' (**đ**) character is difficult because of the very short length of the ascender of the 'd' in some styles, such as Times Roman. In a more leggy font like Palatino (đ), it fits quite well. Someone more versed in font design than the writer should consider this.
   (c) The latter caveat applies even more strongly to the 'ij' characters (**ij** and **IJ**)
2. Not all Adobe fonts have the same characters; this may cause a problem in the future, though it is rather more likely that fonts will *increase* their ranges and acquire, for instance, an ffi ligature (ffi) or a Thorn (**Þ**). Rather more serious is the fact that the thousands of copies of, say, Palatino already in use are different versions, and symbols available on one printer's copy may be absent on another. 'y acute' (**ý**) is a good example of this, not being present in older copies of Adobe TimesRoman.
3. Some of the symbols that we are used to in TEX (such as † and ‡) have been squeezed out of the layout. We need to agree on a useful set of symbols as a companion to the extended TEX layout.
4. Lastly, is this the right approach? It might be more effective to do all the calculation of new composite characters in POSTSCRIPT itself, so that no virtual font was needed at all. We must bear in mind, however, that at some point a font metric file must be created for TEX (possibly generated automatically from information in the POSTSCRIPT font dictionary, which is relatively easy).

Some of these problems are soluble with a little patience; others need some careful expert advice; yet others may need manual intervention for each font. In the worst case, a font design program may have to be brought in to create the missing symbols.

6 Conclusions

The techniques presented in this article illustrate the skeleton of methods whereby fairly wide-ranging changes can be made to the effect of a POSTSCRIPT font, beyond simple remapping. The same technique may be appropriate to create ISO Latin-1 layouts, for instance.

Readers who use their POSTSCRIPT fonts with other software (such as Adobe Type Manager) may

```
pplr0 Palatino-Roman " ECEncoding ReEncodeFont " <ec.enc
ptmro0 Times-Roman  " ECEncoding ReEncodeFont " <ec.enc  ".167 SlantFont"
plcb0 Lucida-Bold <plcb.pfb " ECEncoding ReEncodeFont " <ec.enc
hlcdi40 LucidaFax-DemiItalic  " ECEncoding ReEncodeFont " <ec.enc <hlcdi4.pfb
```

Figure 5: Example additions to `dvips` file `psfonts.map`

```
%
/ECEncoding [          % now 256 chars follow
% 0x00
  /grave /acute /circumflex /tilde /dieresis /hungarumlaut /ring /caron
  /breve /macron /dotaccent /cedilla
  /ogonek /quotesinglbase /guilsinglleft /guilsinglright
% 0x10
  /quotedblleft /quotedblright /quotedblbase /guillemotleft
  /guillemotright /endash /emdash /compoundwordmark
  /perthousand /dotlessi /dotlessj /ff /fi /fl /ffi /ffl
% 0x20
  /visiblespace /exclam /quotedbl /numbersign
  /dollar /percent /ampersand /quoteright
  /parenleft /parenright /asterisk /plus /comma /hyphen /period /slash
```

Figure 6: Start of encoding file for `afm2tfm`

prefer to eschew this whole approach in favour of manipulating a copy of the Type1 font itself. Commercial programs are available which allow you not only to change the encoding of a font, but also to add new ligatures and composite characters in the same way as our virtual fonts do. This would provide PostScript fonts which can be used in any environment directly, and by TeX without virtual fonts.

The programs and header files used to create the examples in this article are available by electronic mail from the author, or from the UK TeX archive at Aston (`ftp.tex.ac.uk`). As this work is extremely derivative, the writer is very grateful to Norbert Schwarz (co-ordinator of the Cork table, and author of the dc fonts), Tom Rokicki and Donald Knuth (whose `afm2tfm` supplied the basis), Amanda Walker (who showed how to create missing characters), Alexander Samarin (some additions to `vpltovpl.c`), Karl Berry, and Berthold Horn (discussions of font encoding problems).

References

[1] Adobe Systems Incorporated 1985 PostScript *Language Reference Manual*, Addison-Wesley, Reading, Massachusetts

[2] Adobe Systems Incorporated 1987 PostScript *Language Tutorial and Cookbook*, Addison-Wesley, Reading, Massachusetts

[3] Bień, J. 1990. 'On standards for computer modern font extensions' Tugboat 11, no. 2, pp. 175–183

[4] Beebe, N. 1990. 'Character set encoding', Tugboat 11, no. 2, pp. 171–174

[5] Knuth, D. 1990. 'Virtual fonts: more fun for grand wizards' Tugboat 11, no. 1, pp. 13–23

◇ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y01 2JR
U.K.
spqr@minster.york.ac.uk

	'0	*'1*	*'2*	*'3*	*'4*	*'5*	*'6*	*'7*	
'00x	`	´	ˆ	˜	¨	˝	°	ˇ	"0x
'01x	˘	¯	˙	¸	˛	,	‹	›	
'02x	“	”	„	«	»	–	—		"1x
'03x	‰	ı			fi	fl			
'04x	␣	!	"	#	$	%	&	’	"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	{	|	}	~	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8x
'21x	Ĺ	Ľ	Ł	Ń	Ň	r	Ő	Ŕ	
'22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	"9x
'23x	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§	
'24x	ă	ą	ć	č	ď	ě	ę	ğ	"Ax
'25x	Í	l’	ł	ń	ň	r	ő	ŕ	
'26x	ř	ś	š	ş	ť	ţ	ű	ů	"Bx
'27x	ÿ	ź	ž	ż	ij	¡	¿	£	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	þ	SS	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 7: Extended TEX layout in Times Roman

	'0	*'1*	*'2*	*'3*	*'4*	*'5*	*'6*	*'7*	
'00x	`	´	ˆ	˜	¨	˝	°	ˇ	"0x
'01x	˘	¯	˙	¸	˛	,	‹	›	
'02x	“	”	„	«	»	–	—		"1x
'03x	‰	I	J	ff	fi	fl	ffi	ffl	
'04x	␣	!	"	#	$	%	&	’	"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	{	|	}	~	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8x
'21x	Ĺ	Ľ	Ł	Ń	Ň	ŋ	Ő	Ŕ	
'22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	"9x
'23x	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§	
'24x	ă	ą	ć	č	ď	ě	ę	ğ	"Ax
'25x	Í	ľ	ł	ń	ň	ŋ	ő	ŕ	
'26x	ř	ś	š	ş	ť	ţ	ű	ů	"Bx
'27x	ÿ	ź	ž	ż	ij	¡	¿	£	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	đ	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ss	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 8: Extended TEX layout in Lucida Bright derived small caps

A LaTeX macros to set up accented characters and ligatures

Practical use of the extended layout requires that we redefine some TeX or LaTeX macros which deal with accented characters. This is addressed perhaps more thoroughly in support files distributed with the New Font Selection Scheme for LaTeX, but a skeleton code is presented below:

```
%-------------------------------------------------
% ecacc.tex
%
% set up composite characters and accents
% assuming TeX Extended Layout (Cork September 1990)
%
% many combinations are included as ligatures
% in the virtual font
%
% Sebastian Rahtz October 10th 1990; August 20th 1992; Sep 9th 1992
%-------------------------------------------------
% some obvious ASCII characters used for other purposes
\chardef\%='\%
\chardef\&='\&
\chardef\#='\#
\chardef\$='\$
% a group of common extended characters
% this could probably be usefully expanded
\chardef\ss='377%  germandbls
\chardef\ae='346%  ae
\chardef\oe='367%  oe
\chardef\o='370%   oring
\chardef\AE='306%  AE
\chardef\OE='327%  OE
\chardef\O='330%   Oslash
\chardef\i='031%   dotless i
\chardef\j='032%   dotless j
\chardef\aa='345%  aring
\chardef\AA='305%  Aring
\chardef\l='252%   lslash
\chardef\L='212%   Lslash
%
%\def\_{\leavevmode\kern.06em\vbox{\hrule width.3em}}
\chardef\_='\_
% some miscellaneous symbols
\chardef\pounds='277%
\chardef\guillemotleft='023%
\chardef\guillemotright='024%
\chardef\guilsinglleft='016%
\chardef\guilsinglright='017%
\chardef\quotesinglbase='015%
\chardef\quotedblbase='022%
\chardef\S='237%                    section mark
%
% but these are not in the new font:
% need agreement on a symbol font layout
% to include all this sort of thing
%\def\dag{{\symfont \char'207}}%     dagger
%\def\ddag{{\symfont \char'210}}%    doubledagger
%\def\P{{symfont\char'266}}%         paragraph mark
%
\ifx\protect\undefined\let\protect=\relax\fi
\def\pd#1{\oalign{#1\crcr\hidewidth.\hidewidth}}
\def\d{\protect\pd}%                dotunder accent
\def\pb#1{\oalign{#1\crcr\hidewidth
   \vbox to.2ex{\hbox{\char'055}\vss}\hidewidth}}
```

```
\def\b{\protect\pb}%                barunder accent
%
% accents for manual combination
\def\pc#1{\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent'013#1%
 \else{\ooalign{\hidewidth\char'013\hidewidth\crcr\unhbox\z@}}\fi}
\def\c{\protect\pc}%                cedilla
\def\`#1{{\accent'001 #1}}%             grave
\def\'#1{{\accent'000 #1}}%             acute
\def\v#1{{\accent'007 #1}}\let\^^_=\v%  hacek
\def\u#1{{\accent'010 #1}}\let\^^S=\u%  breve
\def\^#1{{\accent'002 #1}}\let\^^D=\^%  circumflex
\def\.#1{{\accent'012 #1}}%             dotaccent
\def\H#1{{\accent'005 #1}}%             hungarumlaut
\def\~#1{{\accent'003 #1}}%             tilde
\def\"#1{{\accent'004 #1}}%             dieresis
\def\=#1{{\accent'011 #1}}%             macron
%
% how do you specify ogonek ?
%
\def\acute{\mathaccent"7001 }   % acute
\def\grave{\mathaccent"7000 }   % grave
\def\ddot{\mathaccent"7004 }    % dieresis
\def\tilde{\mathaccent"7003 }   % tilde
\def\bar{\mathaccent"7009 }     % macron
\def\breve{\mathaccent"7008 }   % breve
\def\check{\mathaccent"7007 }   % caron
\def\hat{\mathaccent"7002 }     % circumflex
\def\dot{\mathaccent"700A }     % dotaccent

% upper and lowercase codes
\lccode223=255 % Germandbls
\uccode223=223
\uccode255=223
\lccode255=255
%  etc etc; rest omitted
```

B Creating new characters in PostScript

When I originally worked to produce EC-style PostScript fonts, I created a tiny four-character font which contained the characters commonly missing from PostScript fonts. This approach was clumsy, and in daily use of the fonts, I have simply ignored the missing characters, or used a font (like Lucida Bright) which had a full set of glyphs.

The code for deriving the characters is listed here out of interest:

```
%%BeginDocument: texchars.pro
%%
%% create a small new font with some extra characters
%% S Rahtz December 1990; stolen from Amanda Walker's font for TeX
%%
/TeXZZencoding 256 array def
0 1 255  { TeXZZencoding exch /.notdef put } for
TeXZZencoding dup 0
/ff put dup 1 /ffi put dup 2 /ffl put dup 3 /dotlessj put pop
/MakeTeXChars {
20 dict begin
        /FontType 3 def
        /FontMatrix [.001 0 0 .001 0 0] def
        /FontBBox [0 0 1000 1000] def
        /FontName exch def
        /BFont exch def
        /BaseFonts [ BFont findfont 1000 scalefont ] def
        /Encoding TeXZZencoding def
```

```
        /String 1 string def
        /CharProcs 5 dict def
        CharProcs begin
                /.notdef { } def
                /ff {
                        (ff) stringwidth exch 50 sub exch
                        0 0 moveto (ff) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (f) show
                } def
                /ffi {
                        (f\256) stringwidth exch 50 sub exch
                        0 0 moveto (f\256) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (\256) show
                } def
                /ffl {
                        (f\257) stringwidth exch 50 sub exch
                        0 0 moveto (f\257) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (\257) show
                } def
                /dotlessj {
                        (j) stringwidth
                        0 0 moveto (j) false charpath flattenpath pathbbox
                        6 copy setcachedevice newpath
                        0 0 moveto (\365) false charpath flattenpath pathbbox
                        newpath -1000 -1000 moveto
                        dup -1000 exch lineto
                        1000 exch lineto
                        1000 -1000 lineto
                        closepath clip
                        pop pop pop
                        0 0 moveto (j) show
                } def
        end
        /BuildChar {
                exch begin
                BaseFonts 0 get  setfont
                dup Encoding exch get
                dup CharProcs exch known
                        { CharProcs exch get exch pop exec }
                        { pop String exch 0 exch put
                          String stringwidth
                          newpath 0 0 moveto String false charpath
                          flattenpath pathbbox
                          setcachedevice
                          0 0 moveto
                          String show
                        } ifelse
                end
        } def
        currentdict
end
dup /FontName get exch definefont pop
} def
%%EndDocument
```