Documentation for the use of the \charsubdef primitive

June 6, 1990

Prof. Michael J. Ferguson

INRS-Telecommunications

3 Place du Commerce

Verdun, Quebec H3E 1H6

Canada

INTRODUCTION:

The modifications described here allow the new TeX 2.993+ to use the current fonts and enhance it to allow for the hyphenation of words with characters that do not explicitly appear in the font. The modification consists of

* char_sub.ch ... a change file for the program TeX. This change file introduces two new TeX primitives \charsubdef and \tracingcharsubdef.

* a set of TeX macro definitions that redefine the accent macros, such as \^e , so that the appropriate 8bit code is used if the substitution list for the character exists but uses TeX's original definitions if it does not. In addition, there are a set of macros that allow for the inputting of hyphenation patterns and exceptions

with accented characters encoded in TeX's backslash form ... ie both ü and \"u are acceptable input forms. The use of these macros allows for almost 100% compatibility with Multilingual TeX.

This note consists of a description and syntax of the new primitives and macros, a discussion of the appropriate use of the primitives with a warning of potential surprises, and a short section on modifications required to allow a Multilingual TeX environment to be compatible with the new TeX.

The basic idea of the extension is to define a two character sequence for each letter and to rebuild that character, if it does not exist in the font, just before it is sent out to the dvi file. The idea is quite powerful, but at the moment is restricted to just a single accent and base letter. When TeX needs a letter for spacing and such, it uses the base letter in the list.

THE PRIMITIVE \charsubdef

The heart of the extension is the new primitive \charsubdef. This new primitive defines the substitution sequence for the extended character.
The syntax is as follows:

```
     \charsubdef <ext char> [=] <accent> <base char>
```

The = is optional and each of the other arguments are character numbers. The syntax is similar to the TeX primitive \chardef. TeX allows for a character number to be expressed in octal, hexadecimal, decimal or symbolically. Thus the e-circumflex, ê, may be described in the four following equivalent ways ... assuming the IBM-PC internal code.

```
octal:    '212
hex:      ^^8a
decimal:  138
symbolic: `\ê
```

Thus the \charsubdef definition for ê would be

```
\charsubdef `\è = '022 `\e
```

Note that the code '022 is that of the accent in the font and not the code for ` . The symbolic forms are used whenever possible to avoid error. An equivalent form would be

```
\charsubdef '212 = '022 '145
```

The TeX "internal" encoding used should be the same as that of the equivalent character in the 256 code font. Thus when TeX checks for the existence of the character, it will indeed be checking for the same character. The "Compatibility " macros allow for a very simple mapping from the keyboard code to the internal code.

THE PRIMITIVE \tracingcharsubdef

The \charsubdef for any given character may be modified, much like most other TeX parameters, while the document is being processed. Since the character is rebuilt just before it is sent out to the dvi file, the \charsubdef actually in force at that time is the one used. It is recommended that all the \charsubdef(initions) be made in the format file. However, since it is possible to modify a \charsubdef dynamically, setting the primitive \tracingcharsubdef non-zero will report everytime that \charsubdef is used. This should help in determining whether there has been a change in a \charsubdef before a particular \shipout. In addition, if \tracinglostchars >=100, then everytime that a character is rebuilt using a \charsubdef, it is reported in the log file.

THE COMPATIBILITY MACROS:

The "Compatibility" macros define an equivalent and efficient macro "inverse" for each of the characters defined with a \charsubdef. This "inverse" is used when determining whether an accented character built using a macro sequence such as \"u should be replaced by its equivalent 8bit internal code or built using the accent primitive. For example, \"u would usually be replaced by ü while \^t would not.

\csubinverse{ext char}{accent macro invocation letter}{base char}

Unlike the \charsubdef, this macro takes actual characters as arguments. The {ext char} and the {base char} are as before but the {accent macro invocation char} is a string representation of the keyboard character that appears in the macro such as \'e rather than the font code for the accent. For the acute accent this is {@ac@}. Thus the inverse for è would be

```
         \csubinverse è{@ac@}e
```

The definition of \csubinverse is

```
\def\csubinverse #1#2#3{\expandafter\def\csname #2#3\endcsname{#1}}
```

Accent Macro Definitions

These macros check to see if a \csubinverse has been defined for a particular sequence. If the inverse exists, the equivalent extended character code is substituted. If it does not exist, the accent primitive is used in its normal fashion. The accent sequences such as \^e, are used in exactly the same manner as in ordinary or Multilingual TeX. An example of a definition for the acute accent \' is

```
\def\'#1{{\expandafter\ifx\csname '#1\endcsname\relax
           {\accent19 #1}\else\csname '#1\endcsname\fi}}
```

Because some characters, such as ˜ , are normally active in TeX and have special meanings, both the original inverse and the accent forms are defined slightly differently. Thus a \˜ used Spanish for \˜n, is defined as

```
\def\~#1{{\expandafter\ifx\csname @til@#1\endcsname\relax
           {\accent'176 #1}\else\csname @til@#1\endcsname\fi}}
```

The complete set of macros is included in the file compatible.tex.

SOME SAMPLE EXTENDED CHARACTER DEFINITIONS

The complete list of extended character definitions in the ISO Latin 1 internal coding is in the file extdef.tex. This file inputs compatible.tex before it processes the character codes.

An example for the ä and Ä

```
\catcode`\ä=11 \lccode`\ä= `\ä \charsubdef `\ä = '177 `\a
\csubinverse ä{@um@}a
\catcode`\Ä=11 \lccode`\Ä= `\ä \charsubdef `\Ä = '177 `\A
\csubinverse Ä{@um@}A
```

The \catcode`\ä=11 defines ä as a letter, the \lccode`\ä= `\ä defines the \lccode to be itself while \lccode`\Ä= `\ä defines the \lccode for the uppercase Ä to be the same as the lowercase ä. The \charsubdef `\ä = '177 `\a is the actual charsub form. The \csubinverse ä{@um@}a defines the inverse for the macro checks. Note that the second argument is @um@.

Some characters appear "normally" as an extended character but are not accessed via the accent macros. An example of that is the Swedish Å, the capital "circle A" which is accessed in TeX with a \AA macro. The extended character equivalent is essentially the same as in Multilingual TeX, except that we must explicitly declare the extended character active. Thus we have

```
\catcode`\Å=\active
\def Å{\AA }
```

To disable a previous \charsubdef it is necessary to define it as a pair of zero values. Thus

```
\charsubdef '321 = '000 '000  % N tilde -- disabled
```

removes the \charsubdef for the character whose internal (ISO) code is '321. This happens to be the N-tilde, Ñ. The information in the log file, in conjunction with \tracinglostchars, and \tracingcharsubdef is sufficient to discover the internal coding of any character.

Inputting Hyphenation Patterns and Exceptions.

TeX has very limited macro capabilities when processing the \patterns

primitive. It is able to process a \csname <...> \endcsname but not any tests. The key to having the hyphenation proceed is to replace the accented characters in the patterns and exceptions with their extended character code. Since there is no guarantee that extended character codes will be consistent across all TeX installations, the convention is to input the extended characters in their macro form. Thus è is input as \‘e and î as \^\i.
The macro \accenthyphcodes performs this magic. This form is complete for French but might require extensions for other languages. The definition is

```
\gdef\accenthyphcodes{
\def\oe{^^[} % \oe
\def\i{^^P}
\def\'##1{\csname @ac@##1\endcsname}
\def\‘##1{\csname @gr@##1\endcsname}
\def\v##1{\csname @v@##1\endcsname}
\let\^^_=\v
\def\u##1{\csname  @u@##1\endcsname}
\let\^^S=\u
\def\=##1{\csname @eq@##1\endcsname}
\def\^##1{\csname @hat@##1\endcsname}
\let\^^D=\^
\def\.##1{\csname @dot@##1\endcsname}
\def\H##1{\csname @H@##1\endcsname}
\def\~##1{\csname @til@##1\endcsname}
\def\"##1{\csname @um@##1\endcsname}
\let\c@@=\c
\def\c##1{\csname c@##1\endcsname}
}
```

```
\gdef\spechyphcodes{}
```

The \gdef\spechyphcodes{} is input for compatibility with Multilingual TeX.

Thus the French hyphenation patterns, identical to those in multilingual TeX would be input as

```
% french hyphenation patterns
\begingroup
\language=1
\input frhyph \relax
\endgroup
```

Multilingual TeX also redefines the hyphenation exception primitive as follows:

```
\let\h@yphenation=\hyphenation
\def\hyphenation#1{{\spechyphcodes\accenthyphcodes \h@yphenation{#1}}}
```

A form with or without the \spechyphcodes is required for this extension to process hyphenation exceptions correctly.

MODIFICATIONS REQUIRED FOR OLD MULTILINGUAL TeX Systems.

Old Multilingual TeX systems are completely compatible with the new \charsubdef TeX with a few exceptions. These exceptions are

* \dischyph is not implemented. This means that words that include explicit discretionaries will not be hyphenated.

* Current implentations of TeX 3 restrict the number of trie_ops to 256 per language. Multilingual TeX had a restriction on the total only. This restriction is being relaxed in some newer implementations, for example, PCTeX allows for 512. Extensions of this sort will almost certainly become essential.

Multilingual TeX permanently set all internal codes above 127 to be \active. This meant that to define a character such as ê only required the following:

```
\def ê{\^e}
```

In the new version, you must explicitly define the ê active. Thus you must use

```
\catcode ‘\ê = \active
\def ê{\^e}
```

instead. Any character so defined will not be allowed in a macro name. This restriction is currently true for Multilingual TeX.

COMMENTS:

The primitive \charsubdef used to be called \charsublist. The change to \charsubdef was made to emphasize its similarity to \chardef in syntax. In addition, the current version (June 1990) introduces the new primitive \tracingcharsubdef and fixed a number of bugs in the implementation of the character rebuilding. In particular, diacritics that appear below the letter, such as the cedilla in C, are recognized by their zero height and not raised over capital letters. TeX's accent routine is considerably more magical.