

# X-Symbol Manual

---

Semi-WYSIWYG for LaTeX, HTML and other “token languages”  
Edition 4.5.1 (XEmacs), for X-Symbol 4.5.1, May 2003

by Christoph Wedler

---

Copyright © 1998-2003 Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Free Software Foundation.

# 1 Introduction

When you edit LaTeX, HTML, BibTeX or T<sub>E</sub>Xinfo sources in Emacs, package X-Symbol provides some kind of WYSIWYG by using real characters for tokens like `\oplus` or `&trade;`. It also provides various input methods to insert these characters. Thumbnails for included images and real super-/subscripts and are also supported.

## 1.1 X-Symbol's Copying Conditions: GPL

(This text is stolen from the T<sub>E</sub>Xinfo manual, Edition 4.0).

The programs currently being distributed that relate to X-Symbol include Emacs Lisp files and X11 font files. These programs are *free*; this means that everyone is free to use them and free to redistribute them on a free basis. The X-Symbol related programs are not in the public domain; they are copyrighted and there are restrictions on their distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of these programs that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to X-Symbol, that you receive source code or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of the X-Symbol related programs, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the programs that relate to X-Symbol. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the licenses for the programs currently being distributed that relate to X-Symbol are found in the General Public Licenses that accompany them.

## 1.2 Brief Summary of X-Symbol

- X-Symbol provides a **minor mode** which make use of characters in the Latin-1, Latin-2, Latin-3, Latin-5, and Latin-9 font (179 chars + 294 char aliases), the Adobe symbol font (109 chars) and the xsymb1 font (165 chars, distributed with the package). Additional fonts could be used easily.
- These characters are used in the buffer to represent **tokens** (e.g., T<sub>E</sub>X macros, SGML entities, more “token languages” could be added easily) in the file. The *conversion* is done automatically when visiting the file, saving the buffer and turning the minor mode on/off.
- Defines 8 **input methods** for these characters: *Menu*, *Grid* (selecting a character with the mouse), *Keyboard*, *Context* (replace/modify similar-looking char sequence), *Electric* (automatic replace), *Quail* (a Mule input method), *Token* (replace token by corresponding char), *Read Token* (completing minibuffer input of token).
- Offers some **info** in the echo area for these characters (e.g., that the character under point represents the TeX macro `\leadsto` and that the macro is defined in LaTeX package ‘`latexsym.sty`’).

- Allows to use a **8bit file encoding** which is different from your "normal" 8bit file encoding, e.g., you can visit T<sub>E</sub>X files with `\usepackage[latin5]{inputenc}` even if you normally use a Latin-2 font.
- Provides a kind of “**poor man’s Mule**” when running on an XEmacs without Mule support: it can *display* more than 256 characters via `font-lock` and removes most annoyances resulting from the fact that, without Mule support, many “X-Symbol characters” are actually a sequence of two chars.
- Provides fonts for single-line innermost **super-** and **subscripts** to be displayed with per-buffer control. The invisible part, like `<sub>` in HTML, is revealed at point.
- Displays thumbnails for **images** at the end of image insertion commands with per-buffer control (e.g., `\includegraphics{file}` in L<sup>A</sup>T<sub>E</sub>X, `<img src=file>` in HTML). They show a scaled-down version of the included image files (using `convert` from **ImageMagick**). A single mouse click on the image or command invokes the image editor for the corresponding image file.
- It *does not* and *will not* provide commands to hide (more or less) uninteresting parts of your document or fontify them differently. This is more the task of the corresponding major mode or `font-lock`, e.g., `font-latex`. (I admit, the support of super- and subscripts might let you think that this is a good point for the todo list of package X-Symbol.) Using `outline-minor-mode` or folding might also be an alternative.

If you prefer a more WYSIWYG-like document processor, you should probably use LyX or GNU TeXmacs. Here are some reasons why you would use Emacs/XEmacs with package X-Symbol instead:

- You have complete control over the L<sup>A</sup>T<sub>E</sub>X source. X-Symbol supports more characters.
- You can read any L<sup>A</sup>T<sub>E</sub>X source and you write normal L<sup>A</sup>T<sub>E</sub>X code, i.e., package X-Symbol does not use any special format.
- It also supports HTML and T<sub>E</sub>Xinfo documents and BibT<sub>E</sub>X entries.
- You can use your favorite editor, i.e., Emacs or XEmacs.

## 1.3 About this Manual

Apart from this manual, there are two other sources of information about X-Symbol:

- The web pages of X-Symbol provide a summary of X-Symbol, including some screen shots. You are strongly encouraged to read them carefully. They probably provide enough info for the standard user and can be found at:  
<http://x-symbol.sourceforge.net/>
- The online help for commands (functions) and user options (variables) is quite technical. It is shown during customization and when using Emacs’ Help menu.

This manual is somewhere in between: it more detailed than the web pages and less technical than the online help. For example, when explaining some functionality, it states the default behavior, gives an impression of what can be customized, and it even lists all related user options, but it does not describes the technical format of possible values of each option.

If you want to learn something about X-Symbol’s internals, e.g., if you want to define your own token language, see [Chapter 7 \[X-Symbol Internals\]](#), [page 51](#).

This manual does not explain Emacs in general or some optional programs used by this package such as `convert` (used to produce the image thumbnails). It also includes no installation instructions for those programs and the author of this package will not help you with the installation of those programs (sorry for that).

You do not have to learn this manual by heart before sending a question to the maintainer of X-Symbol, but you should give the impression that you really have tried to find the necessary information yourself and spend some time making your report precise. Before sending a problem report, please read [Section 8.5 \[Bug Reports\]](#), page 68.



## 2 Installation

The short version of the installation instructions for package X-Symbol on XEmacs is: uncompress & extract the *binary distribution* in directory ‘`~/xemacs/xemacs-packages/`’, add (`x-symbol-initialize`) to your ‘`~/xemacs`’ and install ImageMagick for the image support (unless you want to get a warning).

Please check the web page *additionally* to the sections here for the installation instructions for package X-Symbol on Emacs.

The rest of this chapter contains the long version. I recommend that you read this chapter completely after a short test of X-Symbol, especially if you have customized your Emacs more or less heavily or if you get some problems.

### 2.1 Requirements

This development version of package X-Symbol works with Emacs-21.1 or higher, and XEmacs 20.4 or higher (XEmacs-21.1.9 is strongly recommended, [Section 8.4.1 \[FAQ XEmacs Core\]](#), [page 64](#)), with or without Mule support.

X-Symbol should work with all window systems Emacs is running under (Mac is not tested and might not work). Under X, no restrictions apply. Under Windows with Emacs, images will not be displayed (they are not yet supported by Emacs under Windows). Under Windows with XEmacs, X-Symbol just supports a limited number of characters (Latin-1, Latin-5, and half the math symbols) and no super- and subscripts, due to missing MS-Windows fonts (see [Section 9.2.2 \[Wishlist Fonts\]](#), [page 77](#)). Under a character terminal, X-Symbol just supports Latin-1 characters only, no super- and subscripts and no images.

This package requires package `font-lock` (distributed with Emacs and XEmacs), the use of package `lazy-shot` is recommended, see [Section 2.6.2 \[Syntax Highlighting Packages\]](#), [page 8](#).

If you want to see the images at the end of image insertion commands, install `convert` from [ImageMagick](#), see [Section 2.5 \[Installing Image Converter\]](#), [page 7](#). They show a scaled-down version of the included image files.

If you want to produce the Info files yourself (they are included in the binary distribution), you need `makeinfo`, Version 1.68 or higher. If you want to produce a PS file from the manual, you need `texi2dvi`. If you want to produce an HTML version of this manual, you need `texi2html`, Version 1.62 or higher. See [Section 2.10 \[Installing Manual\]](#), [page 13](#).

### 2.2 Put the Files into your Home Directory

If you use Emacs, please check the [web pages of X-Symbol](#).

In this section, we assume that you want to install the binary distribution (also called the binary tarball) of package X-Symbol in your home directory. To install it somewhere below the XEmacs root (it might be already there), see [Section 2.3 \[System-wide Installation\]](#), [page 6](#). If you use the source distribution, you should know what to do instead.

In directory ‘`~/xemacs/xemacs-packages/`’, run

```
zcat x-symbol-pkg.tar.gz | tar xvf -
```

Remember that `tar` does not overwrite write-protected files.

X-Symbol’s ‘`pcf`’ files and font directory must be world-readable since you do not own the X11 font server process. You are on the safe side, if you run

```
chmod -R a+rx ~/.xemacs/xemacs-packages
```

If package X-Symbol has been installed system-wide and you install a newer version in your ‘`~/.xemacs/xemacs-packages/`’, you get a warning during XEmacs’ startup (autoload error: already loaded). You can safely ignore this warning, but there is unfortunately no good way to get rid of it. Yes, XEmacs’ packaging system is excellent, but there is still a place for improvements. . . .

Before XEmacs-21.0: the user package directory was ‘`~/.xemacs/`’ instead of ‘`~/.xemacs/packages/`’; also: delete and recompile the ‘`.elc`’ files.

## 2.3 System-wide Installation: Put the Files into the XEmacs Directory

You can skip this section if you have installed X-Symbol in your home directory according to the previous section.

If you install package X-Symbol system-wide, use ‘`default.el`’ and ‘`xemacs/site-packages/`’ whenever ‘`~/.emacs`’ and ‘`~/.xemacs/xemacs-packages/`’ are mentioned in the previous or following subsections. ‘`xemacs/site-packages/`’ is the directory of independent packages for XEmacs.

Under XEmacs-21, you can uncompress and extract the tarball by

```
M-x package-admin-add-binary-package (RET) dir/x-symbol-pkg.tar.gz
```

Then, ‘`xemacs/`’ is the default directory of buffer ‘`*Package Output*`’ (use `C-x C-f` in that buffer to see it). It might be ‘`/usr/local/lib/xemacs/xemacs-packages/`’ (the first element in variable `late-packages`).

Under XEmacs-20, ‘`xemacs/`’ might be ‘`/usr/local/lib/xemacs-version/`’. Here, you have to uncompress and extract the tarball as described in [Section 2.2 \[Installing Files\]](#), [page 5](#). You also have to load the autoload file explicitly by putting the following line into file ‘`site-start.el`’:

```
(load "xemacs/lisp/x-symbol/auto-autoloads")
```

I would appreciate if you would set the following variables:

**x-symbol-installer-address**

Please set this variable to your email address to catch problems which could be solved locally. In your private ‘`~/.emacs`’, you might want to set this variable to `nil`.

**x-symbol-package-url**

If you have a local copy of the web pages (see [Section 2.10 \[Installing Manual\]](#), [page 13](#)), set this variable to the corresponding URL.

## 2.4 Make XEmacs Initialize X-Symbol During Startup

Put the following into your ‘`~/.emacs`’ (or ‘`~/.xemacs/init.el`’):

```
(x-symbol-initialize)
```

Basically, that’s it! If your XEmacs runs on a different machine, check [Section 2.7 \[Installing Fonts\]](#), [page 10](#).

If you get a warning about X-Symbol not being able to deduce a default encoding (or about limited support with XEmacs under Windows or a character terminal, [Section 2.1 \[Requirements\]](#), [page 5](#)), set the default coding (see [Section 3.2.1 \[Default Coding\]](#), [page 15](#)) by putting the following in front of the line above:



```
(setq x-symbol-default-coding 'iso-8859-1)
```

When running Emacs under a character terminal, you might need to use the following (with or without X-Symbol):

```
(unless window-system (standard-display-european 1))
```

If your character terminal does not support Latin characters, there is no reason to use package X-Symbol. In this case, use the following instead:

```
(when window-system (x-symbol-initialize))
```

The initialization can be controlled by the following variable:

**x-symbol-initialize**

By default, package X-Symbol does a full initialization. This includes an integration with some packages, see also [Section 2.6 \[Package Integration\]](#), page 7.

If you use a B/W monitor and XEmacs/no-Mule, it might be necessary to remove the font properties of any face which is used on regions with X-Symbol characters: `isearch`, `highlight`, `primary-selection`, `secondary-selection`, `paren-match`, `paren-mismatch`, `paren-blink-off`, `underline`. I.e., for each *face*, use:

```
(remove-specifier (get (get-face 'face) 'font))
```

## 2.5 Installing the Image Converter from ImageMagick

Program `convert` from ImageMagick is used to display images at the end of image insertion commands. The images show a scaled-down version of the included image files.

While the installation of `convert` is optional, you get a warning if `convert` is not found on your system or if there is no image format supported by both `convert` and Emacs. Set variable `x-symbol-image-converter` to `nil` if you don't want to get the warning.

On Unix, `convert` must be in your `$PATH`. On Windows, it is assumed to be found at `'C:\ImageMagick\convert'`. If this is not the case, you have to customize the variable `x-symbol-image-convert-program`.

Check <http://www.imagemagick.org/> for the installation instructions. Run `'convert -h'` and `'convert -list Format'` (in newer versions of ImageMagick) in your shell to check whether the installation of ImageMagick was successful. If you have problems, check the ImageMagick web page for FAQs and mailing lists.

If you do not have a truecolor device (i.e., just 256 colors), package X-Symbol uses `convert` with a colormap by default (see [Section 5.2.2 \[Image Conversion\]](#), page 34). You might create and use your own colormap instead. It should be tuned to include the colors you use in Emacs anyway, i.e., the face colors.

## 2.6 Package Integration

You might skip this section when trying package X-Symbol the first time. Nevertheless, I strongly recommend to read this section if you have customized your Emacs more or less heavily or if you get some problems.

Some features of X-Symbol work by hooking itself into existing functions of Emacs or related packages via predefined hooks. A potential problem arises if your customization or other packages use the same hooks, or if other packages assume these hooks not to be used, e.g., some packages assume the buffer contents to contain the same characters as the corresponding file.

This section lists some special adaptation for other packages (everything is fine if you do not use these packages). It also lists potential problems in combination with other packages. If you discover some problems in combination with other packages, please let me know.

### 2.6.1 LaTeX Packages

Objectives: relate positions in buffer to positions in file, do conversion in master/slave buffers, preserve highlighting, improve input methods and other things.

**auctex** Use Version 9.9c or higher, which includes `texmathp`. There is some special X-Symbol adaptation for AucTeX:

- X-Symbol supports AucTeX’s multifile documents: it respects the variable `TeX-master` when searching for the file encoding (see [Section 3.2.2 \[File Coding\]](#), [page 16](#)) and when converting image files with relative names (see [Section 5.2.1 \[Image Display\]](#), [page 34](#)).
- X-Symbol supports AucTeX’s region commands: it ensures that characters in ‘`_region.tex`’ buffer are converted according to the parent buffer. Initialization changes `TeX-region-hook`. Requires AucTeX, v9.8a or higher.
- X-Symbol’s input method Electric (see [Section 4.8 \[Input Method Electric\]](#), [page 30](#)) with token language `tex` uses package `texmathp`.
- AucTeX’s math mode commands also inserts X-Symbol characters (see [section “Mathematics” in AUCTEX](#)). Initialization sets `LaTeX-math-insert-function`. Requires AucTeX, v9.8a or higher.
- If TeX displays an error message, it also displays the context of the error position. AucTeX uses the context to set point to this position when `M-x TeX-next-error` is invoked. The former context are characters in the file, the latter characters in the buffer, X-Symbol provides the translation. Initialization changes `TeX-translate-location-hook`.

**bib-cite** Use Version 3.0 or higher. Initialization of package X-Symbol changes the installation of package `bib-cite` to make X-Symbol’s decoding not overwrite `bib-cites` highlighting of `\cite` and friends.

**preview-latex**

TeX’s error positions are also used by package `preview-latex`, which was clever enough to reuse the above mentioned hook of AucTeX. Unfortunately, that hook is ... and does not allow a fast translation of error positions, so `preview-latex` allows to provide better variants of functions in that hook. X-Symbol’s variant is `x-symbol-tex-preview-locations`.

**reftex** Use Version 3.26 or higher. For a workaround for some minor annoyances with the combination RefTeX/X-Symbol/Multifile Document, see [section “Problems and Work-arounds” in RefTeX User Manual](#). By default, the initialization of package X-Symbol makes RefTeX’s label creation use the nicer Ascification of package X-Symbol (see [Section 5.4 \[Ascii Representation\]](#), [page 38](#)) by setting `reftex-translate-to-ascii-function`.

**whizzytex**

Use the newest.

### 2.6.2 Syntax Highlighting Packages (font-lock and add-ons)

Objectives: start highlighting after conversion. Highlighting is needed for super- and subscripts and when using XEmacs without Mule support.

**fast-lock**

I recommend to use package `lazy-shot` instead. By default, the initialization of package X-Symbol sets `fast-lock-save-faces` to `nil` to make package `fast-lock` work with X-Symbol.

**font-latex**

I suggest to set `font-lock-maximum-decoration` to value `t`, 2 or higher if you do not want to use super- and subscripts in arguments of `\label` and friends. See [Section 8.4.5 \[FAQ Stupid Subscripts\]](#), page 66.

**font-lock**

Is required by this package (see [Section 3.5 \[Role of font-lock\]](#), page 22). I strongly recommend *not* to turn on font-lock in *any* mode hook, set `font-lock-auto-fontify` to `t` instead (this is the default, anyway). See also `lazy-shot`.

If you turn on font-lock in a mode-hook, visiting a file would become slower, since X-Symbol mode is usually turned on *after* the functions in the mode hook have been run, i.e., the fontification is getting useless if the tokens are automatically decoded.

**lazy-lock**

From XEmacs-20.3 on, the successor is called `lazy-shot`.

**lazy-shot**

Is strongly recommended.

### 2.6.3 File I/O Packages

Issue: compression, encryption and so on can be seen as some kind of conversion. When doing multiple conversion, the sequence matters.

**ange-ftp** See also `efs` and `jka-compr`.

**comint** The default installation makes `comints` in-/output use X-Symbol's conversion function. If you set variable `comint-input-sender`, set it before initializing package X-Symbol.

**crypt**

**crypt++** I recommend to use package `jka-compr` instead. See [Section 8.2 \[Spurious Encodings\]](#), page 63. See [Section 8.3 \[No Encoding\]](#), page 64. If you use `crypt` or `crypt++` and the character `alpha` looks like `'\233a'` after `save-buffer`, set this variable to `slowest`. See [Section 9.3 \[Open Questions\]](#), page 79.

**efs** XEmacs' version of `ange-ftp`. See also `jka-compr`.

**iso-cvt** There is no need to use it. Package X-Symbol already provides the conversion between Latin-1 characters and "TeX macros". Package X-Symbol does not provide the German and Spanish conversion tables, though.

**iso-sgml** There is no need to use it. Package X-Symbol already provides the conversion between Latin-1 characters and "SGML entities". See [Section 2.6.4 \[Miscellaneous Packages\]](#), page 10, package `psgml-html`.

**jka-compr**

Can be used with package X-Symbol, preferred to `crypt`. The following is absolutely necessary (with or without using package X-Symbol, at least in older Emacsen): load `jka-compr` after `efs/ange-ftp`!

**latin-unity**

This XEmacs package can be used with package X-Symbol, functionality is already provided by X-Symbol for Latin-{1,2,3,5,9} characters: remapping (see [Section 3.2.7 \[Char Aliases\]](#), page 20) and recoding (see [Section 3.2.2 \[File Coding\]](#), page 16). Has some safe-encoding mechanism, but the test comes currently too early (see [Section 9.2.3 \[Wishlist Emacs\]](#), page 78).

**ucs-tables**

The Emacs minor modes `unify-8859-on-decoding-mode` and `unify-8859-on-encoding-mode` can be used with package X-Symbol.

**vc**

If you use package `crypt`, `vc-next-action` and friends encode characters to tokens. See [Section 8.2 \[Spurious Encodings\]](#), page 63.

## 2.6.4 Miscellaneous Packages

**abbrev**

On XEmacs without Mule support, I recommend to set variable `words-include-escapes` to `t`. See [Section 8.1 \[Nomule Problems\]](#), page 63.

**completion**

Should work with X-Symbol (earlier version of X-Symbol had problems with input method token).

**desktop**

XEmacs' version (an old one) does not save its `~/.emacs.desktop` files with a coding system. Emacs' version save it with an incorrect coding system. Thus, strings which contain X-Symbol's private characters might get corrupted. See also package `session` below.

**flyspell**

Should work apart from the general problem of `ispell`.

**func-menu**

Should work with X-Symbol.

**ispell**

The package `ispell` assumes the buffer contents to be the same as the file contents and does not provide any hook to fix this. This should be fixed in `ispell`, see [Section 9.2.3 \[Wishlist Emacs\]](#), page 78. See [Section 8.4.11 \[FAQ Spell Check\]](#), page 67.

Use a future version (hopefully v3.4). Includes special X-Symbol initialization/handling and defines additional token languages. See [Section 6.6 \[External Languages\]](#), page 50.

**psgml-html**

`psgml-html`: Do not set `html-auto-sgml-entity-conversion` to non-nil. See [Section 2.6.3 \[File IO Packages\]](#), page 9, package `iso-sgml`.

**session**

Use Version 1.5a or higher. If strings in this file should always be read correctly, you should put `(x-symbol-init-input)` into your `~/.emacs`; otherwise strings containing X-Symbol's private characters read from the `~/.session` file might look funny. See also package `desktop` above.

**x-compose**

All characters from `x-compose` are also supported by package X-Symbol. Thus, I recommend to use `(multi-key)` instead `C-=` when running under XEmacs without Mule support. See [Section 4.1 \[Introducing Input Methods\]](#), page 25.

## 2.7 Installing Additional Fonts

You don't have to install X-Symbol fonts in usual circumstances (with the binary distribution, Emacs runs on the same machine, you are happy with the default fonts).

If your Emacs runs on a different machine, please follow the steps 5 and 6 below or read the next section.

If you want to install additional fonts (since the binary distribution contains only a limited selection of fonts and font sizes), please follow the following sequence which worked for me (on

SunOS 5.4-5.6/Solaris). If you have to do s.th. (completely) different on your system, please let me know—I will include your hints.

If you are lost with the following instructions, use the standard fonts from the binary distribution. (Sorry, I do not have to time to answer general Unix font questions. Or to be more exact, I'm not an expert in this area. . . . Nevertheless, if you have a clearer explanation for the installation sequence below, please send me a patch to 'man/x-symbol/x-symbol.texi'.)

1. Find the font which you want to replace by checking fonts with the X11 program `xfontsel` or `xfd`. The bad news is that there is no general way to say which character belongs to which font. My only goal was to use standard fonts whenever possible; the rest belong the the `xsymb1` font (which I have designed). If you want to use a font as an alternative to another font, it must have the same charset registry-encoding.
2. Find the '.bdf' files of your preferred fonts in your file system or by Internet search engines like Google. The source distribution of package X-Symbol contains '.bdf' files for additional fonts sizes of all fonts except the `xsymb1` font (see [Section 9.2.2 \[Wishlist Fonts\]](#), page 77). There are two categories of '.bdf' files. The first category contains files for fonts which are already installed; the files are needed to create and install the super- and subscript versions. Copy these files to '~/.xemacs/xemacs-packages/etc/x-symbol/origfonts/'. The second category contains files for fonts which are not installed. Copy these files to '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/'.
3. In file '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/Makefile', change variables `ORIGBDFS` for the first category and `BDFS` for the second category accordingly.
4. In directory '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/', execute `make makedirs`, and `make pcfs`. You need GNUs `make` and `perl`, Version 5 (or higher).
5. If your Emacs runs on a different machine or if you want to use the fonts outside Emacs, too, add X-Symbol's fonts to your font path by inserting the following in your '~/.xsession' (X11 startup file).

```
xset +fp ~/.xemacs/xemacs-packages/etc/x-symbol/pcf/
```

For a system-wide installation, you might want to add this directory to the system-wide font path instead.

If your system doesn't have `xset`, you should copy all '.pcf' files (compiled fonts) from '~/.xemacs/xemacs-packages/etc/x-symbol/pcf/' into directory '/usr/lib/X11/fonts/75dpi/' (Slackware distribution) and run '`mkfontdir 75dpi`' in that directory.

6. You are on the safe side if you restart X11 after this.
7. Set the Emacs Lisp variables which define the fonts. See [Section 2.9 \[Installing Fonts Lisp\]](#), page 12.

## 2.8 Installing Fonts for Exceed (X-server on Windows)

If your X-server on Windows is Exceed and if you have configured Exceed to use the "native window manager" for your Unix screens, you must install the X-Symbol fonts on Windows. The following works with Exceed 6.0 & NT 4.0 and Exceed 7.0 & Windows 2000:

1. In Unix, edit file '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/makesub' to limit the shift for superscript to 3 points:

```
%supoffs = ('08',3, 10,3, 12,3, 14,3, 16,3, 18,3, 24,3);
```
2. Then, execute `make makedirs`, and `make gens` in '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/'. If you have problems, please read the previous section.
3. In Exceed's configuration window, click on (Font) to open Window 'Font Settings'. In this window, click on (Select All), then on (Compile Fonts...).

4. Copy X-Symbol's 'bdf' files in '~/.xemacs/xemacs-packages/etc/x-symbol/fonts/' and '~/.xemacs/xemacs-packages/etc/x-symbol/genfonts/' to a Windows directory and select this directory in the Exceed Window 'Compile Fonts'. Click on Compile.
5. In Window 'Font Settings', click on Font Database... In this window, click on Add... Enter the output directory from the previous step as the 'Font Directory' and xsymb as the 'File Name (\*.fdb)'. Click on OK.
6. You might want to rearrange the sequence of Font DB files to let files '75dpi' appear prior to files '100dpi', because X-Symbol's fonts are designed for 75dpi.
7. In Window 'Font Database', click on Rebuild Database... and then on OK.

Note: *Windows NT 4.0 will crash (bluescreen) if you use fonts compiled by Exceed from the 'pcf' files or if you missed step 1, i.e., limiting the superscript shift!* With Exceed 7.0 & Windows 2000, there is no crash, but these fonts cannot be displayed.

If you use XEmacs with Exceed as your X-server on Windows, X-Symbol cannot warn you about undefined fonts, because XEmacs in general cannot recognize in that case, whether a font exists.

## 2.9 Lisp Coding when Using Other Fonts

Package X-Symbol needs to know which fonts to use for the X-Symbol characters and super- and subscripts. It also must interact with package font-lock to display them (see [Section 3.5 \[Role of font-lock\]](#), page 22).

If you have installed additional fonts (see [Section 2.7 \[Installing Fonts\]](#), page 10) for use with package X-Symbol, you might have to change the following variables:

```
x-symbol-latin1-fonts
x-symbol-latin2-fonts
x-symbol-latin3-fonts
x-symbol-latin5-fonts
x-symbol-latin9-fonts
x-symbol-xsymb0-fonts
x-symbol-xsymb1-fonts
```

The value of each variable consists of three elements: one for the normal text, one for subscripts and one for the superscripts. Each element is a list of fonts which are tried in order—the first which exists on your system is used.

If you change the values of one of these variables, do only specify the same charset registry—encoding (e.g., 'adobe-fontspecific') as specified by the fonts in the default value of this variable.

```
x-symbol-font-sizes
```

Here you can specify the sizes for all fonts in the above mentioned variables. The value consists of regular expressions matching font names and numbers which replace all occurrences of '%d' in the names.

E.g., if you prefer larger fonts, you might want to insert the following into your '~/.emacs':

```
(setq x-symbol-font-sizes
      '(18 ("_su[bp]-" . 14) ("\\'-etl-" . 16)))
(setq x-symbol-xsymb0-fonts
      '(("adobe-symbol-medium-r-normal-***%d0-***-adobe-fontspecific"
         "-xsymb-xsymb0-medium-r-normal--%d-%d0-75-75-p-85-adobe-fontspecific")
        ("adobe-symbol_sub-medium-r-normal-***%d0-***-adobe-fontspecific"
         "-xsymb-xsymb0_sub-medium-r-normal--%d-%d0-75-75-p-74-adobe-fontspecific")
        ("adobe-symbol_sup-medium-r-normal-***%d0-***-adobe-fontspecific"
         "-xsymb-xsymb0_sup-medium-r-normal--%d-%d0-75-75-p-74-adobe-fontspecific")))
```



The first assignment changes the font sizes, the second makes X-Symbol using the original Adobe symbol font instead of my minor modification (appearance) of it. The xsymb1 font will be scaled, which might not look nice (see [Section 8.4.6 \[FAQ Font Size\]](#), page 66).

You might want to change the following variables:

#### **x-symbol-latin-force-use**

Package X-Symbol defines Latin characters even when the corresponding fonts are missing (this can be changed by this variable). Characters for the symbol fonts are only defined if the corresponding fonts are available.

#### **x-symbol-mule-change-default-face**

Package X-Symbol does not change the fonts of pre-defined Mule charsets (this can be changed by this variable). Thus, the variables from [Section 2.9 \[Installing Fonts Lisp\]](#), page 12 might have no influence if Emacs already has defined fonts for the corresponding charsets.

## 2.10 Installing Info, Postscript and HTML Files

To create the info files, execute `make info` in directory `~/ .xemacs/xemacs-packages/man/x-symbol/` of the distribution. It requires `makeinfo`, Version 1.68 or higher. This should not be necessary if you use the binary distribution of package X-Symbol.

If no entry for X-Symbol is automatically added to the info directory listing, add the following line to `~/ .xemacs/xemacs-packages/info/dir`:

```
* X-Symbol::      Semi WYSIWYG for LaTeX, HTML and other "token languages"
```

Optionally, you might want to create a printed document from the `TeXinfo` file. Execute `make ps` in directory `~/ .xemacs/xemacs-packages/man/x-symbol/` of the distribution. It requires `texi2dvi`.

Optionally, you can create an online manual for a web browser by executing `make html` in directory `~/ .xemacs/xemacs-packages/man/x-symbol/` of the distribution. It requires `texi2html`.

All formats of the manual are created by executing `make all`.

## 2.11 Checking the Correct Installation of Package X-Symbol

After having completed the installation, exit and restart Emacs.

- Type `M-x show-message-log` to check whether you got problems so far, e.g., whether errors occurred when loading a file. If you do, identify and correct the offender.
- Type `M-x x-symbol-grid` in buffer `*scratch*`. If you get the Grid but if you see less characters than you see on the web page of package X-Symbol, you have decided to use other fonts but failed to install them correctly. This is also mentioned in buffer `*Warnings*`. See [Section 2.7 \[Installing Fonts\]](#), page 10.
- Move your mouse pointer to any X-Symbol character in buffer `*X-Symbol Grid (x-symbol charsym)*`, press the right mouse button and initialize successively all token languages.
- Again, type `M-x show-message-log` to check whether you got problems so far, e.g., whether errors occurred when loading a file. If you do, identify and correct the offender.
- If buffer `*Warnings*` does not exist in the buffer menu, everything is fine. So is (for me as the author of package X-Symbol), if 'X-Symbol' is not mentioned there. If there is a warning with `'no valid image converter'`, you have forgotten to install ImageMagick (see [Section 2.5 \[Installing Image Converter\]](#), page 7).





## 3 Concepts of Package X-Symbol

This chapter describes the concepts of package X-Symbol. It contains quite a few forward references to feature which are based on these concepts, such as [Chapter 4 \[Input Methods\]](#), [page 25](#), and [Chapter 5 \[Features\]](#), [page 33](#).

### 3.1 Token Language

As mentioned in the overview, “X-Symbol Characters” in the buffer are represented by “tokens” in the file. The correspondence between these is determined by the *token language* which is in close relation to the major mode of the current buffer. E.g., character `alpha` stands for `\alpha` in LaTeX buffers.

For details of predefined token languages “TeX macro” (`tex`), “SGML entity” (`sgml`), “BibTeX macro” (`bib`), and “TeXinfo command” (`texi`), see [Chapter 6 \[Supported Languages\]](#), [page 41](#).

The token language determines the conversion between X-Symbol characters and tokens (see [Section 3.2 \[Conversion\]](#), [page 15](#)), the input methods (see [Chapter 4 \[Input Methods\]](#), [page 25](#)), and various other features (see [Chapter 5 \[Features\]](#), [page 33](#)).

The token language is defined by the following buffer-local variable:

`x-symbol-language`

Token language used in current buffer. You can set this variable in the “local variables list” near the end of the file (see [section “File Variables” in XEmacs User’s Manual](#)), e.g.:

```
%% Local Variables:
%% x-symbol-language: tex
%% End:
```

Package X-Symbol uses a reasonable value according to the major mode and the file name of a buffer if the variable is not already buffer-local. A valid token language is required to turn on X-Symbol Minor mode, see [Section 3.3 \[Minor Mode\]](#), [page 20](#).

A token language must be *registered*, if you want to use it. By default, the above mentioned token languages are registered.

### 3.2 Conversion: Decoding and Encoding

As mentioned, X-Symbol characters in the buffer are represented by tokens in the file. Thus, we need some conversion from tokens to characters, called *decoding*, and some conversion from characters to tokens, called *encoding*.

We have the additional problem that some characters are not only represented by tokens, but also via some 8bit character encoding.

Package X-Symbol supports the following 8bit character encodings: Latin-1 (`iso-8859-1`), Latin-2 (`iso-8859-2`), Latin-3 (`iso-8859-3`), Latin-5 (`iso-8859-9`), and Latin-9 (`iso-8859-15`). It currently supports less encodings with XEmacs on Windows (see [Section 2.1 \[Requirements\]](#), [page 5](#)).

#### 3.2.1 Normal File and Default Encoding

As mentioned, some characters have a 8bit file encoding, and X-Symbol needs to know which 8bit file encoding you use normally when visiting a file and saving a buffer.

With Mule support, Emacs/XEmacs can recognize the *normal file encoding*, also called a coding system (see [section “Recognize Coding” in XEmacs User’s Manual](#)).

Without Mule support, XEmacs can usually only support 8bit characters of one encoding; this encoding corresponds to the charset/registry of your default font. Here, the *normal file encoding* is the default encoding:

#### **x-symbol-default-coding**

The default encoding. The value must be a symbol denoting one of the supported encodings or `nil`. The variable must be set before X-Symbol has been initialized. See [Section 2.4 \[Installing Lisp\]](#), page 6.

The *default encoding* is not only used to determine the normal file encoding without Mule, but also for the following:

- X-Symbol has its own mechanism to recognize a file encoding which only works with a specified default encoding. See [Section 3.2.2 \[File Coding\]](#), page 16.
- The same character can be included in various Latin charsets and X-Symbol needs to know which of the instances (which Emacs views as different characters) to support. See [Section 3.2.7 \[Char Aliases\]](#), page 20.
- Without Mule support, the default encoding is also needed to decide which characters have to be faked by 2 characters internally: exactly the characters in those charsets which do not correspond to the default encoding. See [Section 3.4 \[Poor Mans Mule\]](#), page 22.

To deduce the default value, X-Symbol inspects the Mule language environment and the output of the shell command `locale`, or to be more exact:

```
locale -ck code_set_name charmap
```

Without Mule support, you get a warning if the command does not exist on your system or lists an encoding which is not supported by X-Symbol, such as some Asian encoding. Value `nil` is the same as `iso-8859-1`.

With Mule support, you get a warning if the command lists a supported encoding which is different from the encoding deduced from the Mule language environment. Value `nil` makes sure that X-Symbol file encoding detection (see [Section 3.2.2 \[File Coding\]](#), page 16) only works if Emacs has detected the same encoding; it works like `iso-8859-1` otherwise.

### **3.2.2 File Coding of 8bit Characters**

X-Symbol can use a different encoding for single buffers/files, even if you use X-Symbol on XEmacs without Mule support. To do so, set the following buffer-local variable:

#### **x-symbol-coding**

8bit character encoding in the file visited by the current buffer. Value `nil` represents the normal file encoding (see [Section 3.2.1 \[Default Coding\]](#), page 15).

With Mule support, any value other than `nil` is considered invalid if the normal file encoding is neither the same as this value nor the same as the default encoding. I.e., if your default encoding is `nil`, X-Symbol’s file encoding detection never takes precedence over Emacs’ one, i.e., the normal file encoding.

You can set this variable in the “local variables list” near the end of the file (see [section “File Variables” in XEmacs User’s Manual](#)), e.g.:

```
<!-- Local Variables: -->
<!-- x-symbol-coding: iso-8859-2 -->
<!-- End: -->
```

If the variable is not already buffer-local, a reasonable value is deduced when turning on X-Symbol (see [Section 3.3 \[Minor Mode\]](#), page 20) by searching for some language dependent headers at the beginning of the file:

`x-symbol-auto-coding-search-limit`

X-Symbol usually searches for something like `\usepackage[...]{inputenc}` (see [Section 6.2 \[TeX Macro\]](#), page 41) or `<meta ... charset=...>` (see [Section 6.3 \[SGML Entity\]](#), page 47) in the first 10000 characters.

If you choose not to save a file containing 8bit characters (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17), the file encoding is still important, since the file might contain 8bit characters when you visit it.

If the file encoding is different to the normal file encoding, X-Symbol performs the necessary recoding itself. *Recoding* changes a character with code position *pos* in one charset to a character with the same code position *pos* in another charset. If the normal file encoding is different to the default encoding, X-Symbol also resolves character aliases (see [Section 3.2.7 \[Char Aliases\]](#), page 20).

If you have specified an invalid file encoding (including an encoding different to a non-default normal file encoding), we have the following cases:

- If the normal file encoding is unsupported (any file encoding is invalid in this case) or if the normal file encoding is supported and the file does not contain 8bit characters, we always encode all X-Symbol character (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17). The modeline includes `-i` to represent the file encoding (see [Section 3.3 \[Minor Mode\]](#), page 20), except if the default encoding is `nil`, the normal file encoding is unsupported, and the variable `x-symbol-coding` is not specified.
- If the normal file encoding is supported and the file contains at least one 8bit character, X-Symbol does not touch 8bit characters and never produces them, neither via decoding (see [Section 3.2.4 \[Unique Decoding\]](#), page 18) nor via input methods. The modeline includes `-err` to represent the file encoding (see [Section 3.3 \[Minor Mode\]](#), page 20).

We end with a little example: if your normal file encoding and default encoding is Latin-1, and you visit a file with `\usepackage[latin9]{inputenc}` producing some document containing the Euro sign, you see the Euro character in Emacs when X-Symbol is enabled, but you see the currency character without X-Symbol.

### 3.2.3 Store or Encode 8bit Characters

You can specify that 8bit characters (according to the coding in your file, see [Section 3.2.2 \[File Coding\]](#), page 16), are not encoded to tokens (when saving a file), by setting the following buffer-local variable:

`x-symbol-8bits`

Whether to store 8bit characters when saving the current buffer.

You can set this variable in the “local variables list” near the end of the file (see [section “File Variables” in XEmacs User’s Manual](#)), e.g.:

```
%% Local Variables:
%% x-symbol-8bits: t
%% End:
```

If the variable is not already buffer-local, a reasonable value is deduced when turning on X-Symbol (see [Section 3.3 \[Minor Mode\]](#), page 20) by setting it the the value of `x-symbol-coding`, or searching in the file for 8bit characters:

**x-symbol-auto-8bit-search-limit**

If there is a 8bit character in the file when visiting it, X-Symbol will also store 8bit characters when saving the buffer.

If the file encoding is invalid (see [Section 3.2.2 \[File Coding\]](#), page 16), we always search for 8bit characters in the complete document and set `x-symbol-8bits` accordingly. Then, a non-`nil` value also implies unique decoding (see [Section 3.2.4 \[Unique Decoding\]](#), page 18).

While the variable `x-symbol-8bits` usually only influences the encoding, it also influences the decoding if you choose to decode uniquely (see [Section 3.2.4 \[Unique Decoding\]](#), page 18).

Setting variable `x-symbol-8bits` to `nil` does not necessarily mean that the file will not contain 8bit characters: the characters might have no token representation in the current token language (see [Section 6.5 \[TeXinfo Command\]](#), page 50), or they are glyphs for unused code points in the Latin-3 charset. In both cases, it is unlikely that you have inserted these invalid characters via X-Symbol’s input methods (see [Section 4.1 \[Introducing Input Methods\]](#), page 25), you have probably copied them into the current buffer.

### 3.2.4 Unique Decoding

Token languages might define more than one token representing the same character. When decoding and encoding these tokens, they will be *normalized* to one form, the *canonical representation*. E.g., with language `tex`, visiting a file with tokens `\neq` and `\ne` converts both tokens to character `lessequal`, saving the buffer stores the character as token `\neq` in both occurrences.

It can also happen that a file contains both a 8bit character and a token which would be converted to exactly that character. When saving the file, both characters are either not encoded, or both are encoded to the same token.

Normally, this is no problem. But if you redefine standard `TeX` macros, it certainly could be the case (see [Section 6.2.3 \[TeX Macro Problems\]](#), page 43)! For this reason, package X-Symbol provides the following buffer-local variable:

**x-symbol-unique**

Whether to limit the decoding in such a way that no normalization will happen. That means: only decode canonical tokens, and, if `x-symbol-8bits` is non-`nil` (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17), do not decode tokens which would be decoded to 8bit characters (according to the coding in your file, see [Section 3.2.2 \[File Coding\]](#), page 16).

You can set this variable in the “local variables list” near the end of the file (see [section “File Variables” in XEmacs User’s Manual](#)), e.g., together with a setting for `x-symbol-8bits`:

```
%% Local Variables:
%% x-symbol-8bits: t
%% x-symbol-unique: t
%% End:
```

If the variable is not already buffer-local, a reasonable value is deduced when turning on X-Symbol (see [Section 3.3 \[Minor Mode\]](#), page 20): it will be set to `t` if X-Symbol mode is not automatically turned on.

If the file encoding is invalid (see [Section 3.2.2 \[File Coding\]](#), page 16) and `x-symbol-8bits` is non-`nil` (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17), X-Symbol always uses unique decoding (see [Section 3.2.4 \[Unique Decoding\]](#), page 18).

### 3.2.5 Conversion Commands

First the good news: most of the time, the necessary conversions are performed automatically when you would expect them to be performed:

- Turning X-Symbol minor mode (see [Section 3.3 \[Minor Mode\]](#), page 20) on/off also performs decoding/encoding.
- Saving a buffer where X-Symbol is enabled will encode the characters to tokens in the file (of course, you keep to have the characters in the buffer).
- Inserting a file into a buffer where X-Symbol is enabled will decode the tokens in the inserted region.

Nevertheless, you might want to perform the conversions explicitly in some situations by using one of the following commands (also to be found in the menu):

*M-x x-symbol-decode-recode*

Recode all characters (if necessary) and decode all tokens to characters.

*M-x x-symbol-decode*

Decode all tokens to characters, do not recode characters.

*M-x x-symbol-encode-recode*

Encode all characters in buffer to tokens or recode them.

*M-x x-symbol-encode*

Encode all characters in buffer to tokens. No recoding will be performed since 8bit characters will always be encoded if the file coding is different to the default coding, since `x-symbol-8bits` is relative to the file coding, see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17.

All commands work on the region if it is active, or the (narrowed part of the) buffer if no region is active.

If the file coding is the same as the default coding, the variants with and without recoding (see [Section 3.2.2 \[File Coding\]](#), page 16) do the same. The variants with recodings are the ones used when doing the conversion automatically. The variants without recodings are the ones used when using the special Copy & Paste commands presented in the next subsection.

### 3.2.6 Copy & Paste with Conversion

You probably use X-Symbol, because you want to produce some non-ASCII characters in your final document, but you are not really interested what kind of token you would need to write. (After all, you do not use a hex editor to produce documents using some non-ASCII encoding in the file, since you are not interested in the byte sequence of individual characters.)

Consequently, all editing operations really work on characters, not on the corresponding tokens for the token language of the current buffer. This includes copying and pasting: if you copy the character `plusminus` from a LaTeX buffer to a HTML buffer, you really copy that character and not the three characters of the TeX macro `\pm`.

If you copy text to a buffer where X-Symbol is not enabled, like a mail buffer, that is probably not what you want. Similarly, you would probably like to see the X-Symbol characters for tokens in a text which you have copied from such a buffer. Therefore, X-Symbol provides the following commands (also to be found in the menu):

*M-x x-symbol-copy-region-encoded*

Save the region in the `kill-ring` with all X-Symbol characters encoded like by *M-x x-symbol-encode*, i.e., without recoding.

***M-x x-symbol-yank-decoded***

Reinsert the last text in the `kill-ring` and decode the inserted text like *M-x x-symbol-decode*, i.e., without recoding.

You could get the same result with the usual copy & paste commands and the conversion commands from the previous section (see [Section 3.2.5 \[Conversion Commands\]](#), page 19), but this would clutter the undo information of the current buffer and would require an additional undo operation for the copy.

### 3.2.7 Character Aliases

A *character alias* or *char alias* is a character which is also a character in a font with another registry, e.g., `adiaeresis` is defined in all supported Latin fonts. Emacs distinguish between these five characters. In package X-Symbol, one of them, with `x-symbol-default-coding` (see [Section 3.2.1 \[Default Coding\]](#), page 15 if possible, is supported by the input methods, the other ones are char aliases to the supported one.

The reason is that it would be confusing for the user to choose among different `adiaeresises` and that there are neither different `adiaeresises` in Unicode nor in the token representations of languages `tex` and `sgml`.

8bit characters in files with a file coding `x-symbol-coding` other than `x-symbol-default-coding` are converted to the “normal” form. E.g., if you have a Latin-1 font by default, the `adiaeresis` in a Latin-2 encoded file is a Latin-1 `adiaeresis` in the buffer. When saving the buffer, its is again the right 8bit character in the Latin-2 encoded file.

Thus, in normal cases, buffers do not have char aliases. In Emacs with Mule support, this is only possible if you copy characters from buffers with characters considered as char aliases by package X-Symbol, e.g., from the Mule file ‘`europaen.el`’. In XEmacs without Mule support, this is only possible if you use commands like `C-q 2 3 4`.

If you have char aliases in the current buffer, you might want to use (it is not really necessary, just when searching for characters):

***M-x x-symbol-unalias***

Resolve all character aliases in buffer. If the region is active, only resolve char aliases in the region.

A single char alias before point can be resolved by command `x-symbol-modify-key` and `x-symbol-rotate-key`, see [Section 4.7 \[Input Method Context\]](#), page 29.

The XEmacs package `latin-unity` provides a command to “remap” characters to one character set (if possible). X-Symbol’s unaliasing can be seen as remap operations to a fixed sequence of character sets.

## 3.3 Minor Mode

X-Symbol is a minor mode (see [section “Minor Modes” in XEmacs User’s Manual](#)) which enables the features mentioned in this manual:

- X-Symbol mode is required to do the conversions. Turning the minor mode on/off also includes decoding/encoding (see [Section 3.2.5 \[Conversion Commands\]](#), page 19).
- X-Symbol mode provides the minor mode menu which includes: various commands, commands to insert characters (see [Section 4.4 \[Input Method Menu\]](#), page 26), and entries to change some global and buffer-local variables mentioned in this manual.
- X-Symbol mode is required for most input methods (see [Chapter 4 \[Input Methods\]](#), page 25) and other features (see [Chapter 5 \[Features\]](#), page 33).



With the default installation, X-Symbol mode is automatically turned on when it is appropriate to do so (see below for details). You can control it for individually by the following command:

#### `M-x x-symbol-mode`

Toggle X-Symbol mode. If provided with a prefix argument, turn X-Symbol mode on if the numeric value of the argument is positive, else turn it off. If no token language can be deduced, ask for a token language; if provided with a non-numeric prefix argument (`C-u M-x x-symbol-mode`), always ask.

By default, X-Symbol mode is disabled in special major-modes visiting a file, e.g., `vm-mode` (see [Section 8.4.12 \[FAQ News and Mail\]](#), page 68). Use a prefix argument to be asked whether to turn in on anyway.

Turning X-Symbol mode on requires that you have a valid token language for the current buffer. Since turning X-Symbol mode on also decodes tokens, it is also useful to set the variables which control the conversion (see [Section 3.2 \[Conversion\]](#), page 15).

Since people usually do not want to write some Emacs Lisp functions to do some customizations, X-Symbol provides the following variables which induce X-Symbol to set the necessary buffer-local variables when X-Symbol is turned on:

#### `x-symbol-auto-style-alist`

You can use the major mode and/or the name of the buffer or visited file, and specific functions to set the following variables (if not already buffer-local):

- `x-symbol-token-language` (see [Section 3.1 \[Token Language\]](#), page 15), indicated in the modeline, e.g. `'tex'`,
- `x-symbol-mode`, i.e., whether it is appropriate to turn on X-Symbol mode automatically,
- `x-symbol-coding` (see [Section 3.2.2 \[File Coding\]](#), page 16), indicated in the modeline if different from the default coding, e.g. `'-12'` for Latin-2,
- `x-symbol-8bits` (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17), indicated in the modeline by `'8'`,
- `x-symbol-unique` (see [Section 3.2.4 \[Unique Decoding\]](#), page 18), indicated in the modeline by `'*'`,
- `x-symbol-subscripts` (see [Section 5.1 \[Super and Subscripts\]](#), page 33), indicated in the modeline by `'s'`,
- `x-symbol-image` (see [Section 5.2 \[Images\]](#), page 33), indicated in the modeline by `'i'`,

#### `x-symbol-lang-modes`

Major modes which use token language *lang* by default. See [Chapter 6 \[Supported Languages\]](#), page 41. The languages are checked in registration order (the order shown in the language selection submenus).

#### `x-symbol-lang-auto-style`

Default values for the above mentioned variables `x-symbol-mode`, `x-symbol-coding`, `x-symbol-8bits`, `x-symbol-unique`, `x-symbol-subscripts`, and `x-symbol-image` if not already buffer-local.

#### `x-symbol-auto-mode-suffixes`

Regular expression matching file suffixes to be ignored when checking file names for the derivation above, e.g., extension `'orig'`.

#### `x-symbol-modeline-state-list`

This variable controls the modeline appearance just mentioned.

The menu might also include individual entries for a token language (see [Section 6.2.1 \[TeX Macro Basics\]](#), page 41):

`x-symbol-lang-extra-menu-items`

Extra menu items for each token language *lang* (see [Section 6.2.1 \[TeX Macro Basics\]](#), page 41).

### 3.4 Poor Man's Mule: Running Under XEmacs/no-Mule

Using XEmacs/no-Mule normally means that you are restricted to use not more than 256 different characters in your documents.

Package X-Symbol provides a lot more characters which can also be used with XEmacs/no-Mule. Internally, all X-Symbol characters except the ones of your default font (see [Section 3.2.1 \[Default Coding\]](#), page 15) are represented by two characters, see [Section 7.1 \[Char Representation\]](#), page 51.

This can lead to a lot of problems, which are resolved by the following methods (some annoyances remain, see [Section 8.1 \[Nomule Problems\]](#), page 63) when X-Symbol mode is turned on (see [Section 3.3 \[Minor Mode\]](#), page 20):

- After each editing command, i.e., point movement, deletion of text and insertion of text, package X-Symbol checks whether just one of the two internal characters of an X-Symbol character has been affected.
- Package `font-lock` is used to display these two-character sequences with the correct fonts. The potential problem lies in the set-up of the corresponding font-lock keywords, see [Section 3.5 \[Role of font-lock\]](#), page 22.

`x-symbol-nomule-fontify-cstrings`

Alternatively to enabling `font-lock`, you can run this functions in buffers having the special two-character sequences. With the default installation, this function is run in the selection buffers of package `reftex`.

### 3.5 The Role of font-lock

Package X-Symbol uses package `font-lock` to display super- and subscripts (see [Section 5.1 \[Super and Subscripts\]](#), page 33) and to display its special characters under XEmacs/no-Mule (see [Section 3.4 \[Poor Mans Mule\]](#), page 22). Thus, you should enable `font-lock` in buffers where you want to use X-Symbol (it is by default). See [Section 2.6.2 \[Syntax Highlighting Packages\]](#), page 8.

When X-Symbol mode is turned on, it automatically adds the necessary font-lock keywords to the buffer-local value of `font-lock-keywords` and all font-lock keywords which are commonly used with the current token language.

Setting all font-lock keywords is important since `font-lock` might not yet been turned on or since you might want to change `font-locks` decoration of the current buffer after X-Symbol has been turned on.

Please note that switching the mode by typing *M-x latex-mode* does not set the LaTeX's font-lock keywords! They are set at the end of *C-x C-f*. If you switch the mode, turn on `font-lock` by yourself.

Independently from package X-Symbol, the following command might be useful in some situations:

*M-x x-symbol-fontify*

Refontify buffer.



## 3.6 Character Group and Token Classes

Each X-Symbol character belongs to a *character group*, e.g., `natnums` belongs to `setsymbol`. A character group should consist of similar characters where “similar” means similar meaning, not similar appearance. Two characters which have nearly the same appearance, should be in the same group, though. The group determines:

- The Grid and submenu header under which the character can be found (see [Section 4.5 \[Input Method Grid\]](#), page 27, [Section 4.4 \[Input Method Menu\]](#), page 26).
- The default bindings of characters (see [Section 4.6 \[Input Method Keyboard\]](#), page 28) of some groups.
- Whether to show the context info for a character (see [Section 5.3 \[Info\]](#), page 37).
- The default ASCII representation of a character (see [Section 5.4 \[Ascii Representation\]](#), page 38).
- When using Emacs/XEmacs with Mule support, the syntax of a character (see [section “Syntax” in XEmacs User’s Manual](#)).

The character group is independent from any token language, but is probably somewhat related to some of its *token classes*. For each token language, each character is assigned to a list of token classes, which can be used for the following:

- Information in the echo area (see [Section 5.3 \[Info\]](#), page 37), it could inform users to include a specific LaTeX package when they want to use that character in the document.
- Using a *coloring scheme* when displaying the characters in the echo area (see [Section 5.3 \[Info\]](#), page 37) or the Grid of characters (see [Section 4.5 \[Input Method Grid\]](#), page 27), useful for characters which can just be used in a specific context, like TeX’s math-mode characters.
- Restricting the “electricity” of input method Electric (see [Section 4.8 \[Input Method Electric\]](#), page 30), useful to disable this input methods for TeX’s math-mode characters if we are in text-mode.

The token classes for individual token languages are explained in the corresponding sections of [Chapter 6 \[Supported Languages\]](#), page 41:

`x-symbol-lang-header-groups-alist`

The Grid and Menu headers for each token language *lang*.

`x-symbol-lang-class-alist`

Strings for the character info in the echo area for each token language *lang*.

`x-symbol-lang-class-face-alist`

The coloring scheme for each token language *lang*.



## 4 X-Symbol's Input Methods

An X-Symbol *input method* is a way, provided by package X-Symbol, to insert a X-Symbol character (not in the sense of Mule's "input methods"). For a short overview with screenshots, see the [web pages of X-Symbol](#).

Input methods Token and Electric change the normal way to insert characters a bit. Therefore, they require X-Symbol mode to be turned on and can be turned off explicitly. The other input methods are provided with additional commands and key prefixes, they can also be used in buffers where X-Symbol mode is turned off.

With AucTeX, Version 9.8a or higher, its math mode commands also inserts X-Symbol characters (see [section "Mathematics" in AUCTEX](#)).

### 4.1 Common Behavior of All Input Methods

Input methods normally just inserts *valid characters* which are those characters which have a useful representation in the file:

#### x-symbol-valid-charsym-function

When X-Symbol is turned off, a character is valid if it is an 8bit character according to the value of `x-symbol-default-coding`.

When X-Symbol is turned on, a character is valid if the characters could be encoded to a token in language `x-symbol-language` (see [Section 3.1 \[Token Language\]](#), page 15).

If a buffer is read-only (see [section "Misc Buffer" in XEmacs User's Manual](#)), most input methods push the character to insert onto the kill ring instead. Typing `C-y` lets you then insert the character (see [section "Yanking" in XEmacs User's Manual](#)).

The input methods Keyboard, Menu and Grid (the character selection with `(button2)`) have the same interpretation of the prefix argument:

- With prefix argument '0', do not insert anything, just barf, if the character is not valid.
- With a positive prefix argument, insert a character that many times. Barf, if the character is not valid.
- With a negative prefix argument, insert a character as many times as specified by the absolute value of the prefix argument. A character is also inserted if it is not valid.
- With one or more `C-us` with no digits, insert the token of a language to choose, including "x-symbol charsym" (see [Section 6.1 \[Pseudo Language\]](#), page 41).

Many input commands of package X-Symbol uses the same key prefix in its default binding:

#### x-symbol-compose-key

By default, `C-=` is used as the key prefix. Under XEmacs/no-Mule, you might want to use `(multi-key)` instead:

```
(unless (featurep 'mule) (setq x-symbol-compose-key '(multi-key)))
```

#### x-symbol-auto-key-autoload

Set this to `nil`, if you do not want that pressing `C-=` automatically initializes the input methods.

## 4.2 Input Method Token: Replace Token by Character

If X-Symbol mode is on, input method *Token* automatically replaces the token by the corresponding character when inserting the next character following the token (in some token languages you need the next character to decide whether the token is completed) if it is valid.

The token will be replaced only if the next character has been inserted without prefix argument or with prefix argument 0 (*C-u 0*), the latter will therefore just induce the replacement.

Please note that the token is really replaced by the characters, it is not just `font-lock` which highlights the token to look like a character.

You might want to press *C-/* or *C-x u* to undo the replacement. Input method Token requires X-Symbol mode to be enabled, it can be disabled (and re-enabled) by setting the following variable:

`x-symbol-token-input`

A boolean which can also be changed via the X-Symbol menu.

Individual token language might slightly change the way input method Token works exactly; from the predefined language, it is just `tex` (see [Section 6.2 \[TeX Macro\]](#), page 41).

## 4.3 Input Method Read Token: Minibuffer Completion

You can insert a character by reading the corresponding token in the minibuffer. You are offered completion over the known tokens (see [section “Completion” in XEmacs User’s Manual](#)).

*M-x x-symbol-read-token-direct*

*C-=* `(TAB)` Insert character by selecting a token in the current token language (even if X-Symbol mode is turned off) or an “x-symbol charsym” (see [Section 6.1 \[Pseudo Language\]](#), page 41).

*M-x x-symbol-read-token*

*C-=* `(RET)` Insert character by first selecting the token language and then a token in that language.

Input method Read Token also works if X-Symbol mode is not enabled. It uses the common interpretation of prefix arguments for X-Symbol insert commands, see [Section 4.1 \[Introducing Input Methods\]](#), page 25.

## 4.4 Input Method Menu: Select a Menu Item

If X-Symbol mode is turned on, a *Menu X-Symbol* appears in the menubar (see [Section 3.3 \[Minor Mode\]](#), page 20). It also appears over non-highlighted parts in the Grid and the Key Completions buffer (see [Section 4.5 \[Input Method Grid\]](#), page 27). The menu allows to change buffer-local and global variables (some directly, some via package `custom`). It has a submenu with the most interesting commands of package X-Symbol.

The menu has submenus with commands to insert X-Symbol characters. The submenu headers are the same as the headers in the Grid, see [Section 3.6 \[Char Group\]](#), page 23. The appearance of the menu can be customized:

`x-symbol-local-menu`

With a valid token language, the X-Symbol menu only contains insertion commands for valid characters. The entries are mentioned and sorted according to the token. Otherwise, the X-Symbol menu contains all characters, the entries are mentioned according to their charsym name.

**x-symbol-menu-max-items**

The submenus do not contain more than 30 insertion commands for X-Symbol characters. A submenu is split if necessarily.

Input method Menu also works if X-Symbol mode is not enabled. It uses the common interpretation of prefix arguments for X-Symbol insert commands, see [Section 4.1 \[Introducing Input Methods\]](#), page 25.

## 4.5 Input Method Grid: Choose Highlighted Character

Probably the easiest way to insert a character is by using a *Grid* of characters:

**M-x x-symbol-grid**

**C-- C--** Pops up a buffer displaying X-Symbol characters in a grid like fashion. You can select a character with the mouse or **(RET)**, see below.

In the Grid buffer and the buffer with the possible completions for an X-Symbol key sequence (see [Section 4.6 \[Input Method Keyboard\]](#), page 28), the following commands are used if the mouse pointer is over an highlighted character.

**(button2)**

**(RET)**

**(SPC)**

Insert highlighted character (or character under point, respectively) into the buffer of **point** if **point** is not in the same buffer as the highlighted character. Otherwise, insert the character into the reference buffer, i.e., the buffer where you have invoked the grid or the key completions from. (The reference to the buffer is erased when an X-Symbol character is inserted into any buffer.)

**(button3)**

Pops up a highlight menu where you can select to insert the token of various token languages instead the character itself. In order not to load and initialize all additional token language you have not yet used, the menu offers to do so explicitly for supported (registered) token languages (see [Section 3.1 \[Token Language\]](#), page 15).

Over all non-highlighted parts, the following commands are used:

**(button2)**

Scroll Grid or Key Completions buffer down in upper half of the window and scroll up in the lower half of the window.

**(button3)**

Pops up the X-Symbol menu, see [Section 4.4 \[Input Method Menu\]](#), page 26).

When using the keyboard to select a character, the following command could be useful:

**M-x x-symbol-list-info**

**?**

**h**

**i**

Display info for character under point in echo area.

**M-x x-symbol-list-bury**

**q**

Bury list buffer while trying to use the old window configuration.

You can control the grid by the following variables:

**x-symbol-local-grid**

With a valid token language, the Grid only contains insertion commands for valid characters and might use a coloring scheme. Otherwise, it contains all characters.

**x-symbol-temp-grid**

Inserting an X-Symbol character does not restore the window configuration current before the invocation of the Grid.

**x-symbol-grid-reuse**

Use old Grid when invoking command **x-symbol-grid**, if this is reasonably to do. If **x-symbol-grid** is called with a prefix argument, always create new Grid.

**x-symbol-grid-ignore-charsyms**

The Grid does not contain **nobreakspace**.

**x-symbol-grid-tab-width**

The tab width in the Grid buffer should correspond the font in **x-symbol-heading-face** which is also used as the default font in the Grid buffer.

**x-symbol-heading-strut-glyph**

Use larger interline spacing if a line in the Grid starts with a header.

The headers in the Grid are the same as the submenu headers, see [Section 3.6 \[Char Group\]](#), page 23. Similar looking characters for one headers are grouped together. See [Section 4.7 \[Input Method Context\]](#), page 29.

Input method Grid also works if X-Symbol mode is not enabled. It uses the common interpretation of prefix arguments for X-Symbol insert commands, see [Section 4.1 \[Introducing Input Methods\]](#), page 25.

## 4.6 Input Method Keyboard: Compose Key Sequence

Key sequences starting with **C-=** (see [Section 4.1 \[Introducing Input Methods\]](#), page 25) are used to insert X-Symbol characters, e.g., **C-= ~ >** inserts **leadsto**. The Ascii sequence of the keys after **C-=** look similar to the character which you are going to insert. It is the same as the sequence which is replaced by input method Context, see [Section 4.7 \[Input Method Context\]](#), page 29.

If many characters are represented by the same Ascii sequence, the binding is extended by ‘1’, ‘2’ and so on. If you do not know how to continue your key sequence, the following commands might be useful:

**M-x x-symbol-help**

**C-= zero-or-more-keys** **(help)**

**C-= zero-or-more-keys C-h**

Pops up a buffer displaying possible completions for the key sequence **C-= zero-or-more-keys**. You do not have to type the key sequence again, i.e., **C-= zero-or-more-keys** is also used for the next input.

**C-= zero-or-more-keys** **(button1)**

**C-= zero-or-more-keys** **(button2)**

**C-= zero-or-more-keys** **(button3)**

Use the normal bindings of **(button1)**, **(button2)** or **(button3)**, respectively (see [Section 4.5 \[Input Method Grid\]](#), page 27). The key sequence is not used for the next input.

**C-= zero-or-more-keys M-(prior)**

**C-= zero-or-more-keys M-(next)**

**C-= zero-or-more-keys M-(home)**

**C-= zero-or-more-keys M-(end)**

Execute the commands **scroll-other-window-down**, **scroll-other-window**, **beginning-of-buffer-other-window** or **end-of-buffer-other-window**, respectively. You do not have to type the key sequence again, i.e., **C-= zero-or-more-keys** is also used for the next input.

**x-symbol-temp-help**

Inserting an X-Symbol character restores the window configuration current before the invocation of the Grid.

**x-symbol-map-default-keys-alist**

Defines the bindings mentioned above.

Input method Keyboard also works if X-Symbol mode is not enabled. It uses the common interpretation of prefix arguments for X-Symbol insert commands, see [Section 4.1 \[Introducing Input Methods\]](#), page 25.


## 4.7 Input Method Context: Replace Char Sequence


The idea of the input method *Context* is to replace a sequence of characters by a character which looks similar to the whole sequence. If the sequence consists only of Ascii characters, it is also used for the key bindings, see [Section 4.6 \[Input Method Keyboard\]](#), page 28.

There will be some info in the echo area that the character sequence before point can be replace via input method Context. The following commands are provided:

**M-x x-symbol-modify-key**


**C-**,


**C-=** 

**C-=**  If character before point is an X-Symbol character, “modify” it to an alternative character (if you do it often enough, you are back at your first character). Otherwise replace sequence of characters by a character which looks similar to the whole sequence.

**M-x x-symbol-rotate-key**

**C-**.

**C-=** 

**C-=**  If character before point is an X-Symbol character, “rotate” its “direction” (or change uppercase/lowercase).

Both commands can also be used to resolve a character alias before point, see [Section 3.2.7 \[Char Aliases\]](#), page 20. If the region is active, restrict replacement to use that region since the input method Context only considers the longest sequence of characters with a replacement.

Input method Context can be customized by changing the following variables:

**x-symbol-rotate-prefix-alist**

If you provide a prefix argument to command **x-symbol-rotate-key**, you can specify the direction you want to have: it is according to numerical keypads, e.g., with prefix argument ‘7’ you specify the direction “north-west”.

**x-symbol-rotate-suffix-char**

Command **x-symbol-rotate-key** is also used to “Greekify” the previous character: typing a **C-.** is shorter than a **# C-.,**

**x-symbol-context-ignore**

Constrains whether a context/charsym can be replaced. No constraints by default.

**x-symbol-context-init-ignore**

Contexts starting with a space cannot be replaced. This variable must be set before X-Symbol has been initialized.

## 4.8 Input Method Electric: Automatic Context

The idea of input method *Electric* is to have the input method Context (see [Section 4.7 \[Input Method Context\]](#), page 29) do its replacement automatically. X-Symbol automatically replaces some character sequences of input method Context by the X-Symbol character as soon as the last character in the sequence of the sequence has been pressed.

Input method Electric has nothing to do with the display of super-/subscripts (see [Section 5.1 \[Super and Subscripts\]](#), page 33).

You might want to press *C-/* or *C-x u* to undo the replacement. Input method Electric requires X-Symbol mode to be enabled, it can be disabled (and re-enabled) by setting the following variable:

**x-symbol-electric-input**

A boolean which can also be changed via the X-Symbol menu.

To make input method Electric useful and not annoying, several conditions must be met for X-Symbol to do the auto-replacement:

- Not all contexts will be replaced automatically. E.g., while input method Context allows both pre- and postfixes for accented characters, `:` and `'` only act as prefixes, and `'` and `~` only as postfixes for input method Electric, since these are the combinations where those characters are quite likely not used literally.
- The character must be valid in the current token language, see [Section 4.1 \[Introducing Input Methods\]](#), page 25.
- All characters of the context have been typed without any other command in between, e.g., `->` inserts `arrowright`, `"- left right >` simply inserts `'->'`.
- No prefix argument has been used for any character in the context.
- The electric context must not be a suffix of a longer valid context for another character. E.g., `' ' o` does not insert `'oacute'` because `' ' o` is the context for `ohungarumlaut` (which cannot be inserted by input method Electric).
- It should be “allowed” to change the context to the character via input method Context.
- Individual contexts/charsyms can be disabled by setting the following variables:

**x-symbol-electric-ignore**

The context should neither be `'s'` (this would be annoying when writing English), nor include a space. If you want to disable input method Electric for all accented characters, use

```
(setq x-symbol-electric-ignore
      "[ \t]\\|[A-Za-z][~']\\|[:'] [A-Za-z]")
```

**x-symbol-lang-electric-ignore**

Individual contexts/charsyms can be disabled for each token language *lang*.

## 4.9 Input Method Quail: a Mule Input Method

Another way to insert a characters is by using the Emacs/Mule multilingual text input method “x-symbol” (see [section “Input Methods” in XEmacs User’s Manual](#)).

Again, the Ascii sequence used there is the same as the sequence which is replaced by input method Context, see [Section 4.7 \[Input Method Context\]](#), page 29. A one-letter key sequence is extended by `␣`.

If input method Quail is selected for a buffer, input method Electric (see [Section 4.8 \[Input Method Electric\]](#), page 30) is disabled in that buffer.



## 4.10 Customizing Input Methods

You may safely define key bindings not using the `x-symbol-map` (i.e., starting with `C=`). E.g., for `alpha` on `A-a`, use

```
(global-set-key [(alt a)] 'x-symbol-INSERT-alpha)
```

Please note that the command `x-symbol-INSERT-alpha` is not defined before the main file (`'x-symbol'`) in the package has been loaded (if you really need it, function `autoload` is your friend).

Other possibilities to customize the input methods are by setting the following variables:

`x-symbol-header-groups-alist`

Defines the groups whose characters appear after that header in the Grid and in submenus with that header. See [Section 3.6 \[Char Group\], page 23](#). Extra variables exists for the language dependent Grid and Menu.

`x-symbol-group-input-alist`

`x-symbol-user-table`

These are variables which are used to compute the input definitions. While this kind of indirection might seem complicated to you (it is), it actually ensures consistency across all input methods. See [Section 7.3 \[Defining Input Methods\], page 52](#).

For example, if you prefer `charsym epsilon1` over `epsilon` you might want to use:

```
(setq x-symbol-user-table
      '((epsilon1 t (greek1 "e" nil "epsilon") nil -3000)))
```

`x-symbol-list-mode-hook`

Additional functions to execute after setting up the Grid and Key Completions buffer.

`x-symbol-after-init-input-hook`

You can change the input methods directly by functions in these hooks.



## 5 Features of Package X-Symbol

Package X-Symbol not only provides input methods for X-Symbol characters, it also provides more features which support an easy and comfortable preparation of documents.

### 5.1 Super- and Subscripts

Package X-Symbol displays the characters inside super-/subscript commands in a way to make them look like super-/subscripts. It also marks the super-/subscript command itself as invisible, so you don't see it on the screen. For example, the three characters 'a<sup>2</sup>' in the buffer are *displayed* as an 'a' and a raised, smaller '2'—the '^' is still in the buffer.

Therefore, the display of super- and subscripts has nothing to do with Input Method Electric (see [Section 4.8 \[Input Method Electric\]](#), page 30).

Do not confuse the special Latin characters `twosuperior`, `threesuperior`, `ordfeminine` and `masculine` with the characters '2', '3', 'a' and 'o' when displayed as superscripts. You might notice that the characters look a bit different, but to help you seeing the difference, X-Symbol will display an info in the echo area (see [Section 5.3 \[Info\]](#), page 37) for the special Latin characters when point is before or after the character in question.

X-Symbol only displays the innermost super- and subscripts, since we would need even more additional fonts otherwise. It is also restricted to display single-line super- and subscripts.

The display of super- and subscripts requires `font-lock` to be enabled (see [Section 3.5 \[Role of font-lock\]](#), page 22).

Super- and subscripts are by default enabled if they are defined for the token language and it would be appropriate to turn on X-Symbol automatically for the current buffer (see [Section 3.3 \[Minor Mode\]](#), page 20). They can be disabled (and re-enabled) by setting the following buffer-local variable:

**x-symbol-subscripts**

A boolean which can also be changed via the X-Symbol menu.

As mentioned before, X-Symbol marks the super-/subscript command itself as invisible, except when point is directly before, inside or directly after this command. During the time where this is the case, X-Symbol makes the super-/subscript command reappear and highlights it with pink. This feature can be disabled (and re-enabled) by setting the following variables:

**x-symbol-reveal-invisible**

A boolean which can also be changed via the X-Symbol menu.

**x-symbol-revealed-face**

The face used for the super-/subscript command when revealed.

**x-symbol-idle-delay**

Time in seconds of idle time before revealing invisible characters.

Super-/subscript commands are `^/_` (see [Section 6.2 \[TeX Macro\]](#), page 41) and `<sup>/<sub>` (see [Section 6.3 \[SGML Entity\]](#), page 47):

### 5.2 Images at the end of Image Insertion Commands

Package X-Symbol can display *images* at the end of image insertion commands. They show thumbnails (scaled-down version of the image) for the included image files (using `convert`, see [Section 2.5 \[Installing Image Converter\]](#), page 7). Using the middle mouse button invokes the image editor for the image under the mouse pointer.

### 5.2.1 Display of Images

The display of images is by default enabled if the image commands are defined for the token language and it would be appropriate to turn on X-Symbol automatically for the current buffer (see [Section 3.3 \[Minor Mode\]](#), page 20). It can be disabled (and re-enabled) by setting the following buffer-local variable:

**x-symbol-image**

A boolean which can also be changed via the X-Symbol menu.

Image commands are `\includegraphics` and others (see [Section 6.2 \[TeX Macro\]](#), page 41), and `<img>` (see [Section 6.3 \[SGML Entity\]](#), page 47):

**x-symbol-lang-image-keywords**

The keywords (image commands & arguments) for each token language *lang*.

File names in the image commands must be interpreted correctly. They can be:

- *absolute*, start with `/` or `~`,
- *explicitly relative*, start with `./` or `../`,
- *implicitly relative*, assumed otherwise, e.g., `image.eps`, or
- *special*, like having some special URL prefix like `http:` or `ftp:`.

Relative file names can be relative to some *master directory* (usually the current directory of the file) or to directories in some *search path* (only used with token language `tex`):

**x-symbol-lang-master-directory**

The master directory for each token language *lang*.

**x-symbol-lang-image-searchpath**

The image search path for each token language *lang*. Defaults to the current directory.

**x-symbol-image-searchpath-follow-symlink**

Directories in the search path ending with `//` (double slash) are recursive: all subdirectories not starting with a dot are also included in the search path. If this variable has value `nil` (the default), subdirectories which are symbolic links are not included.

For details, see the section of the individual token languages (see [Chapter 6 \[Supported Languages\]](#), page 41).

### 5.2.2 Image Conversion

The file mentioned inside the image insertion command is not used directly to display the image after the command. The image might be too big, it might use too many colors or the image format might not be supported by Emacs. Therefore, it is converted to an *image cache file*, see [Section 5.2.3 \[Image Caching\]](#), page 35.

**x-symbol-image-max-width**

The image is not wider than 120 points.

**x-symbol-image-max-height**

The image is not higher than 80 points.

**x-symbol-image-convert-colormap**

Colormap used in function `x-symbol-image-convert-colormap` below. A colormap is a normal image whose colors are the only ones used for producing other images. The distribution of package X-Symbol includes two colormaps: `etc/colormap138.xpm` and `etc/colormap66.xpm`.

**x-symbol-image-colormap-allocation**

Package X-Symbol allocates the colors of the colormap at start-up and prevents them to be de-allocated.

**x-symbol-image-converter**

Program `convert` from ImageMagick is used to convert the images (see [Section 2.5 \[Installing Image Converter\]](#), page 7). Set this variable to `nil`, if you don't want to convert images.

The following variables controls the invocation of the program `convert` from ImageMagick:

**x-symbol-image-convert-program**

The name of the program `convert`, it is `'C:\\ImageMagick\\convert'` when running on Windows and `'convert'` otherwise.

**x-symbol-image-convert-file-alist**

Program `convert` needs to be told that `'file.pstex'` is a Postscript file.

The following functions are possible values in `x-symbol-image-converter`:

**x-symbol-image-start-convert-mono**

Produces monochrome images. Used if your device has less than 32 colors.

**x-symbol-image-start-convert-truecolor**

Produce images with original colors. Used if your device has more than 767 colors.

**x-symbol-image-start-convert-color**

Produce images with maximal four colors (just four because different images might use a different sets of colors). Used otherwise without a colormap.

**x-symbol-image-start-convert-colormap**

Produce image with colors from the colormap. Used otherwise with a colormap.

**x-symbol-image-convert-mono-regexp**

Function `x-symbol-image-start-convert-colormap` just produces monochrome images for temporary image cache files (see [Section 5.2.3 \[Image Caching\]](#), page 35) since `convert` is slower when using a colormap.

### 5.2.3 Image Caching

Editing would be extremely slow, if an image cache file would be produced every time an image insertion command has been recognized. Therefore, package X-Symbol uses the following techniques:

- It uses an asynchronous process to create the image cache file. You can edit your file during the conversion.
- It uses a *file cache*: image cache file can be kept for future Emacs sessions.
- It uses a *memory cache*: images from the most common file names are cached in a buffer-local memory cache. The cached is initialized when parsing the whole buffer for image keywords. Rescan the buffer if you want to display the the images of new image files by using the following command:

***M-x x-symbol-image-parse-buffer***

Parse the buffer to recognize image insertion commands. Usually, this is done automatically.

File and memory caching can be controlled by the following variables:

***x-symbol-image-update-cache***

The image cache is automatically updated if it does not exist yet or if it is older than the corresponding image file.

***x-symbol-image-cache-directories***

Cache files for images in your home directory are stored in directory ‘`~/.images/`’, e.g., image ‘`~/d/img.eps`’, is cached in ‘`~/.images/d/img.png`’.

Images outside your home directory are just temporarily cached, or not displayed at all if they cannot be stored in the memory cache.

You could also specify that the cache files uses a relative subdirectory, e.g., that ‘`~/d/img.eps`’ is cached in ‘`~/d/.img/img.eps`’ or that the image is not displayed at all.

***x-symbol-image-temp-name***

Temporary image files are stored in a temporary directory (‘`/tmp/`’) having some unique name. They are not supported on Emacs.

***x-symbol-image-use-remote***

Package X-Symbol only displays images which can be stored in the memory cache. With value `t`, it tries to find the image file during editing (ignoring the search path for speed, though). Editing lines with image files not in the memory cache would be slow, since file accesses are necessary for every command.

The memory cache only stored image file from the current directory or some standard image directories like ‘`figures/`’ (see [Section 6.2 \[TeX Macro\]](#), page 41), or ‘`images/`’ or ‘`pictures/`’ (see [Section 6.3 \[SGML Entity\]](#), page 47). Otherwise, the image file is considered similar to remote files:

***x-symbol-lang-image-cached-dirs***

The directories with images which are stored in the memory cache. Can be separately defined for each token language *lang*.

## 5.2.4 Special Images for Specific Situations

If package X-Symbol cannot display images representing the included image files, it uses special images instead:

- *Remote*: An Escher knot is displayed if the file is remote or if the image cannot be cached in the memory cache, see [Section 5.2.3 \[Image Caching\]](#), page 35.
- *Junk*: A recycle sign is displayed if there is no image converter (see [Section 5.2.2 \[Image Conversion\]](#), page 34), if it should not use a file cache or if the file cache cannot be written.
- *Locked*: A terminal with a lock is displayed if the image cache file cannot be read or written.
- *Design*: An ink pen is displayed if the image file does not exist.
- *Create*: An hour glass is displayed used during the creation of the image cache file, an old image cache is used instead if it exists.
- *Broken*: A tombstone is displayed if the creation of the image cache file has failed.

To customize the glyphs for the special images, use:

`x-symbol-image-data-directory`

Directory of files for the special images.

`x-symbol-image-special-glyphs`

File names of special images and their image format.

### 5.2.5 Image Editor

If you move the mouse pointer to an image insertion command or its image, it is highlighted.

`(button2)` Start image editor for highlighted image. If the image is searched in the searchpath (see [Section 5.2.3 \[Image Caching\]](#), page 35), edit first existing image file. If no image exists, open a new file in the first directory of the searchpath.

`(button3)` Pop up the *image highlight menu*. You can rescan the buffer for image insertion commands (see [Section 5.2.1 \[Image Display\]](#), page 34).

It also displays all directories in the searchpath if the file name is implicitly relative, or the current directory otherwise. Selecting a directory starts the image editor in that directory (relatively to that directory if the file name has a directory part).

`M-x x-symbol-image-editor`

Start image editor. Asks for the image file.

You can control which editor to use:

`x-symbol-image-editor-alist`

Normally, program `display` is used to edit the highlighted image file. But for image names `'file.eps'`, `'file.ps'` or `'file.pstex'`, program `xfig` is invoked with `'file.fig'`. It also uses a scale method, e.g., with `'img.80.eps'`, we edit `'img.fig'` (which should be exported with `scale=80%`).

`x-symbol-image-scale-method`

If a scale method is used for a file name and the file name without extension ends with a dot and two digits, these three characters are removed from the file name.

`x-symbol-image-current-marker`

Directories with an existing image for the specified file name are marked with an `'*`'. The first of these represents the file which is used when pressing `(button2)`.

## 5.3 Info in Echo Area

The echo area (see [section “Echo Area” in XEmacs User’s Manual](#)) is used by X-Symbol to give some information about the character around point, and whether there is a context before point which can be replaced by input method Context (see [Section 4.7 \[Input Method Context\]](#), page 29).

It will be controlled by the following variables (also to be found in the menu):

`x-symbol-character-info`

A three-value variable which controls whether to display some info for the character after or around point. The info for the character after point includes the character itself and the following infos:

- the token of the current language, eventually colored according to some coloring scheme (see [Section 3.6 \[Char Group\]](#), page 23),
- infos using the token classes (see [Section 3.6 \[Char Group\]](#), page 23), which could inform users to include a specific LaTeX package when they want to use that character in the document,

- the codings in which the characters is considered to be a 8bit character (see [Section 3.2.2 \[File Coding\]](#), page 16), and
- the key bindings (see [Section 4.6 \[Input Method Keyboard\]](#), page 28).

#### `x-symbol-context-info`

If X-Symbol mode is on and some conditions are met, display some info for the character which would replace the context before point when pressing `C-`, (see [Section 4.7 \[Input Method Context\]](#), page 29). It can be controlled by the following variables:

##### `x-symbol-context-info-ignore`

The default value `x-symbol-default-context-info-ignore` makes the following variables control whether to display the context info.

##### `x-symbol-context-info-threshold`

The context does not consist of a single character.

##### `x-symbol-context-info-ignore-regexp`

The context does not solely consist of letters.

##### `x-symbol-context-info-ignore-groups`

The context is not replaced by an accented character, see [Section 3.6 \[Char Group\]](#), page 23.

#### `x-symbol-idle-delay`

Time in seconds of idle time before showing the info.

## 5.4 Ascii Representation of Strings

If you want to derive labels from a buffer contents (provided e.g., by Emacs packages `refTeX` or `bibtex`), you need a Ascii representation of strings containing X-Symbol characters. This is provided by the following function:

#### `x-symbol-translate-to-ascii`

Takes a string and returns a string only consisting of Ascii characters.

##### `x-symbol-charsym-ascii-alist`

You might want to define the German way to Asciiify accented characters by:

```
(setq x-symbol-charsym-ascii-alist
      '((adiaeresis . "ae") (Adiaeresis . "Ae")
        (odiaeresis . "oe") (Odiaeresis . "Oe")
        (udiaeresis . "ue") (Udiaeresis . "Ue")))
```

##### `x-symbol-charsym-ascii-groups`

By default, “Ascii”fying accented characters means removing the accents. Other characters have built-in Ascii representation, e.g, `sigma1` has the Ascii representation `'sigma'`.

## 5.5 X-Symbol Package Information

#### `M-x x-symbol-package-info`

Read documentation for package X-Symbol in the info system.

#### `M-x x-symbol-package-web`

Ask a WWW browser to load the URL of package X-Symbol.



***M-x x-symbol-package-bug***

Use this command to contact the maintainer of package X-Symbol *in any case*, e.g., for suggestions, bug and problem reports, see [Section 8.5 \[Bug Reports\]](#), [page 68](#).

Use ***C-u 9 M-x x-symbol-package-bug*** for patches (including corrections of this manual, which are strongly appreciated) and for other messages.

**x-symbol-installer-address**

E-mail address of the person who has installed package X-Symbol system-wide (see [Section 2.3 \[System-wide Installation\]](#), [page 6](#)).

**x-symbol-package-url**

URL of package X-Symbol, used by **x-symbol-package-web**.



## 6 Supported Token Languages

The chapter describe the predefined token language. It also presents the language specific behavior for [Chapter 3 \[Concepts\], page 15](#), [Chapter 4 \[Input Methods\], page 25](#), and [Chapter 5 \[Features\], page 33](#).

### 6.1 Pseudo Token Language “x-symbol charsym”

If no (or an invalid) token language is set for a buffer, the info in the echo area (see [Section 5.3 \[Info\], page 37](#)) for a X-Symbol Character in the buffer (if it exists) uses the name of its *charsym*. In this manual, we actually refer to X-Symbol characters by their charsym name, e.g., `alpha`.

A charsym is a symbol which is used internally to represent a X-Symbol character. Charsyms are used instead characters in all user variables of package X-Symbol.

The highlight menu of the Grid (see [Section 4.5 \[Input Method Grid\], page 27](#)) also offers to insert a charsym name. Charsyms can also be used for input method Read Token, see [Section 4.3 \[Input Method Read Token\], page 26](#).

You cannot use this pseudo language to turn on the X-Symbol minor mode (see [Section 3.3 \[Minor Mode\], page 20](#)), you cannot decode charsyms to their characters, and you cannot encode characters to charsyms.

### 6.2 Token Language “T<sub>E</sub>X macro” (`tex`)

For buffers using the major mode `latex-mode`, `tex-mode` or `plain-tex-mode`, we use token language *T<sub>E</sub>X macro* (`tex`). This language provides the display of super-/subscripts and images. If the buffer visits a file with extension ‘`.tex`’, X-Symbol mode is automatically turned on.

#### 6.2.1 Basics of Language “T<sub>E</sub>X macro”

The standard behavior can be controlled by the following variables:

`x-symbol-tex-modes`

`x-symbol-tex-auto-style`

The variables known from [Section 3.3 \[Minor Mode\], page 20](#). If the buffer visits a file with extension ‘`.tex`’, super-/subscripts and images are displayed, otherwise unique decoding (see [Section 3.2.4 \[Unique Decoding\], page 18](#)) will be used.

`x-symbol-tex-auto-coding-alist`

Used there to automatically deduce the specific encoding of the file (see [Section 3.2.2 \[File Coding\], page 16](#)) if the file visited by the buffer has the extension ‘`.tex`’. It searches for one of the following two strings in the current buffer, including the comment:

```
\usepackage[encoding]{inputenc}
%& -translation-file=ienc
```

where *encoding* should be one of ‘`latin1`’, ‘`latin2`’, ‘`latin3`’, ‘`latin5`’, or ‘`latin9`’, and *enc* should be one of ‘`l1`’ or ‘`l2`’. 8bit characters are not encoded if the file if the search was successful (see [Section 3.2.3 \[Controlling 8bit Coding\], page 17](#)).

`x-symbol-tex-coding-master`

If one of the above strings cannot be found in the current buffer, and the current buffer has a buffer-local string value of `TeX-master`, also search in the file denoted by that value for the strings. (Buffer-local variables will not be inherited.)

The input methods and the character info in the echo area are controlled by:

`x-symbol-tex-header-groups-alist`

We use the standard Grid and Menu headers.

`x-symbol-tex-extra-menu-items`

There is an extra menu item to remove the braces around text-mode letters and other text-mode symbols.

`x-symbol-tex-electric-ignore`

`x-symbol-tex-electric-ignore-regexp`

Input method Electric (see [Section 4.8 \[Input Method Electric\]](#), page 30) is disabled if the character is not of the correct TeX mode, i.e., it only produces a math-mode character in a math area and a text-mode character in a text area (this test requires package `texmathp`, see [Section 2.6.1 \[LaTeX Packages\]](#), page 8). Postfix tilde is not electric, because ‘~’ produces a space in TeX.

`x-symbol-tex-token-suppress-space`

Input method Token (see [Section 4.2 \[Input Method Token\]](#), page 26) only converts a token ending with a control word like `\i`, if the character following the token is no letter. If that token is a text-mode token and a `(SPC)` has been entered without a prefix argument, the `(SPC)` will only perform the replacement, it will not insert a space, i.e., it will act like `C-u 0 (SPC)`.

`x-symbol-tex-class-alist`

`x-symbol-tex-class-face-alist`

Various token classes (see [Section 3.6 \[Char Group\]](#), page 23) are defined. They are used to give some info (see [Section 5.3 \[Info\]](#), page 37) about the characters spacing behavior, which LaTeX packages are necessary to use the character (see [Section 6.2.5 \[TeX Macro Symbols\]](#), page 46), and about the conversion (see [Section 6.2.4 \[TeX Macro Conversion\]](#), page 45). X-Symbol uses blue for text-mode only and purple for math-mode only characters in the Grid (see [Section 4.5 \[Input Method Grid\]](#), page 27 and the character info.

## 6.2.2 Super-/Subscripts and Images in LaTeX

The display of super- and subscripts (see [Section 5.1 \[Super and Subscripts\]](#), page 33) is controlled by:

`x-symbol-tex-font-lock-limit-regexp`

The superscript command `^` and the subscript command `_` is recognized. The argument can be provided with and without braces. The argument should not span more than one line and should not contain a super-/subscript command.

`x-symbol-tex-font-lock-allowed-faces`

The characters ‘`^`’ and ‘`_`’ are not always commands (see [Section 6.2.3 \[TeX Macro Problems\]](#), page 43), e.g., in the argument of `\ref`. X-Symbol uses the usual syntax highlighting keywords to decide whether to recognize these characters as super-/subscript commands: they are commands if they are not highlighted or highlighted with the usual math-mode faces.

This might lead to problems: [Section 8.4.4 \[FAQ No Subscripts\]](#), page 65, [Section 8.4.5 \[FAQ Stupid Subscripts\]](#), page 66. Using `texmathp` (see [Section 2.6.1 \[LaTeX Packages\]](#), page 8) has even more problems:

- The syntax highlighting (which is used for super-/subscripts) would be much too slow.

- With own LaTeX environments, you would need to customize `texmathp`.
- It is actually wrong: whether ‘`^`’ and ‘`_`’ are super-/subscripts commands does not depend on whether we are in TeX’s math mode, it depends on its catcodes (which are changed by commands like `\ref`).

The display of images (see [Section 5.2 \[Images\]](#), page 33) is controlled by:

#### `x-symbol-tex-image-keywords`

The following commands are recognized. Extension *ext* stands for ‘`eps`’ (which is the default extension for both versions of `\includegraphics` if the extension is omitted there), ‘`ps`’, ‘`gif`’, ‘`png`’, ‘`jpeg`’, ‘`jpg`’, or ‘`pdf`’. Options *options* can be omitted with their surrounding brackets or preceding comma, respectively.

```
\input{file.pstex_t}
\includegraphics[options][options]{file.ext}
\includegraphics*[options][options]{file.ext}
\epsfig{file=file.ext,options}
\psfig{file=file.ext,options}
\epsfbox[options]{file.ext}
\epsffile[options]{file.ext}
```

#### `x-symbol-tex-master-directory`

Relative file names (see [Section 5.2.1 \[Image Display\]](#), page 34, explicitly or implicitly) are relative to the directory part of variable `TeX-master` if it is buffer-local and a string. Otherwise, they are relative to the directory of the current file.

#### `x-symbol-tex-image-searchpath`

Files with implicitly relative names are meant to be searched in a search path. It defaults to the list of directories specified by the environment variable `TEXPICTS` or `TEXINPUTS` (see [section “TeX environment variables” in \*Kpathsea Manual\*](#)), extended by ‘`./`’ if necessary.

Each directory in this list is used to expand the file name. The first expansion naming a readable file is used. Relative directories in this list are expanded in the master directory mentioned above.

This mimics the standard behavior of TeX, omitting the “built-in” directories of the search path (see [section “Path sources” in \*Kpathsea Manual\*](#)).

#### `x-symbol-tex-image-cached-dirs`

The file name in the image command should not have a directory part or the directory part should be ‘`figures/`’ if the image should be cached in the memory cache.

## 6.2.3 Problems with TeX Macros

Like with other token languages, the conversion between characters and TeX macros induce the problem that we have two conflicting requirements: we would like X-Symbol not to change the file when visiting and saving a file, and we would like X-Symbol to use characters for all corresponding macros. See [Section 3.2.4 \[Unique Decoding\]](#), page 18.

The additional problem with TeX macros is that there is no fixed and simple definition of TeX macros, and many users have their personal TeX style, while many users are probably not aware that the style also influences TeX’s typesetting:

- The tokens in TeX are not ended by a dedicated character (like SGML entities are ended by ‘`;`’). Instead, we need the next char to decide whether a macro ends, which would be no problem if TeX would have a character which has no meaning except separating tokens

(like space in most programming languages). Unfortunately, this is not the case: after an *control word* (an all-letter macro), a space has no meaning, but it does produce a space in the output after characters and other macros, except in math mode.

During decoding, a text-mode control word has to be replaced either with its trailing spaces or not be replaced at all. Since the number of spaces can vary and X-Symbol does not remember the original  $\text{\TeX}$  sequence of a character, X-Symbol would change the file if it would use characters for all sequences.

- During encoding, a space after a character in the buffer must produce a space in the document output, since users normally do not care whether the character is represented by a control word or not. Let us assume that we (Bavarians) want to produce the output ‘ $\text{\text{Ma}\ss\ Bier}$ ’.
  - Many people would use ‘ $\text{\text{Ma}\ss\ Bier}$ ’. This is (almost ever) fine in text mode, but a ‘ $\backslash$ ’ in math mode is not ignored (whereas the spaces after characters are). If we have text- and math-mode control word, we have a problem, since math-mode detection cannot work properly without  $\text{\TeX}$  processing.
  - Many people would use ‘ $\text{\text{Ma}\ss\{ } Bier}$ ’. This has less problems and is therefore used by X-Symbol. The ‘ $\{ }$ ’ at the end of the control word is not used if the character is not followed by a space, e.g., to produce ‘ $\text{\text{Straße}}$ ’, we use ‘ $\text{\text{Stra}\ss e}$ ’. Consequently, ‘ $\text{\text{Ma}\ss\ Bier}$ ’ in the file would be decoded to ‘ $\text{\text{Ma}\ss\ Bier}$ ’, which would be encoded to the original sequence in the file.
  - Some people would always use ‘ $\{ }$ ’ after a text-mode control word, even it is not followed by a space, like ‘ $\text{\text{Stra}\ss\{ }e}$ ’. This is wrong, since it breaks ligatures and kerns. For example, compare the output of ‘ $\text{\L V}$ ’ with ‘ $\text{\L\{ }V}$ ’ using ‘ $\text{\text{T1}}$ ’ font encoding.
  - Up to Version 4.1, X-Symbol surrounded a text-mode control word with braces, like ‘ $\text{\text{Stra}\ss\{ }e}$ ’. This was probably even worse than always adding ‘ $\{ }$ ’ at the end of the control word. It was used, because it is required by Bib $\text{\TeX}$  (see [Section 6.4 \[BibTeX Macro\]](#), page 49). Unfortunately, Bib $\text{\TeX}$  sends this bad sequence directly to La $\text{\TeX}$ , but this has nothing to do with X-Symbol.
- The accented characters are not represented by one tokens in  $\text{\TeX}$ . Most people use ‘ $\backslash"a$ ’ to produce an ‘ä’, while some use ‘ $\backslash"a$ ’. X-Symbol uses the former, it does not even decode the latter automatically. Up to Version 4.1, X-Symbol used ‘ $\backslash"a$ ’, having the same problems as using ‘ $\text{\text{Stra}\ss\{ }e}$ ’.
- Around a dozen characters can be produced by more than one  $\text{\TeX}$  macro, like  $\backslash\text{neq}$  and  $\backslash\text{ne}$ . Here, X-Symbol decodes both forms, because it is probably a bad idea to redefine standard  $\text{\TeX}$  macros. This will not be done with in style files (see [Section 3.2.4 \[Unique Decoding\]](#), page 18).
- In  $\text{\TeX}$ , you can change the lexer on the fly, i.e., in a strict sense, any conversion is unsafe without  $\text{\TeX}$  processing. Since the most likely change is to change the catcode of the character ‘@’ to a letter (used in La $\text{\TeX}$ ’s style files), this character is considered a letter by X-Symbol. This means that although both ‘ $\backslash\text{ss @}$ ’ and ‘ $\backslash\text{ss@}$ ’ usually produce the same output, only the first is decoded to ‘ $\text{\text{B@}}$ ’.
- In  $\text{\TeX}$ , the definitions of macros can also change on the fly i.e., in a strict sense, any conversion is unsafe without  $\text{\TeX}$  processing. X-Symbol assumes that you do not do something like that except as done by the standard La $\text{\TeX}$   $\backslash\text{verb}$  command, and the `verbatim` and `tabbing` environments.

## 6.2.4 The Conversion of T<sub>E</sub>X Macros

The T<sub>E</sub>X macros for Latin characters are according to the LaTeX package ‘inputenc.sty’, v0.97+. Package X-Symbol uses U00B5 for \mathmicro, not for \mu, though! See [Section 9.2.4 \[Wishlist LaTeX\]](#), page 78.

It is assumed that you do not redefine standard T<sub>E</sub>X macros like \ne (see [Section 6.2.4 \[TeX Macro Conversion\]](#), page 45), if you do so, you should better use unique decoding (see [Section 3.2.4 \[Unique Decoding\]](#), page 18).

The encoding of characters to T<sub>E</sub>X macros works as follows:

- If the character is preceded by an odd number of backslashes, insert a space before the character.
- Accented characters are encoded without braces, e.g., we encode ‘ç’ to ‘\c c’. Accents are encoded with braces, e.g., we use ‘\c{ }’ and ‘\u{ }’.

Additionally, the encoding of characters to T<sub>E</sub>X macros which are *control words* (all-letter macros), or whose T<sub>E</sub>X representation ends with a control word (like ‘\’\i’) works as follows:

- If the character is followed by a letter, replace the character by the macro and insert a space.
- If the macro is a text-mode macro and followed by one or more blanks, replace the character and insert ‘{ }’.
- Otherwise, just replace the character.

The decoding of T<sub>E</sub>X macros which are control words to characters works as follows:

- If the macro is a text-mode macro and followed by ‘{ }’ which is followed by a blank, replace the macro and delete the braces.
- If the macro is a text-mode macro and followed by one or more blanks, we have the following rule:
  - If we have exactly one blank, the blank is a space, and it is not followed by a ‘%’ (comment character), replace the macro by the corresponding character and delete the space. (The character following the space must be a letter with unique decoding, see [Section 3.2.4 \[Unique Decoding\]](#), page 18.)
  - Otherwise, do *not decode* the macro!
- Otherwise, just replace the macro.

To clarify, *letter* means ‘A’-‘Z’, ‘a’-‘z’, or ‘@’, *blank* means a space, newline or the end of the buffer (therefore, the last character in the buffer is always followed by a blank).

There are three control words which are both text-mode and math mode macros: \ldots, \vdots, and (by accident) \angle. They are all treated like math-mode characters, but their minibuffer info (see [Section 5.3 \[Info\]](#), page 37) includes ‘gobbles space’ (spaces in the buffer after the character have no impact on the document),

Additionally, the following commands and environments are processed during decoding (but we are just looking for strings, i.e., they are also processed in comments):

### x-symbol-tex-verb-delimiter-regexp

If the command \verb is found, its argument is not decoded if it is delimited by one of the following characters: ‘-’, ‘!’, ‘#’, ‘\$’, ‘&’, ‘\*’, ‘+’, ‘/’, ‘=’, ‘?’, ‘^’, ‘|’, or ‘!’.

### x-symbol-tex-env-verbatim-regexp

The contents of the verbatim environment is not decoded. To produce accented characters inside this environment, use the LaTeX package ‘inputenc.sty’.



**x-symbol-tex-env-tabbing-regex**

Inside a `tabbing` environment, the macro sequences starting with `\‘`, `\’`, `\=` and `\-` are not decoded. It is probably better (with or without X-Symbol) to use the LaTeX package `inputenc.sty` or to the `Tabbing` environment, to be found in the CTAN archives.

During encoding, these commands and environments are not respected, since it does not make any sense to have X-Symbol’s private characters in the TeX file.

You might want change the conversion between characters and tokens in language `tex` by changing:

**x-symbol-tex-user-table**

You can define you own tokens for X-Symbol characters. E.g., if you like to have the command `\sqrt` represented by a character (shadowing the entry for `\surd`), add the following to your `~/emacs`:

```
(setq x-symbol-tex-user-table '((radical (math special) "\sqrt")))
```

## 6.2.5 Extra Symbols of Language “TeX Macro”

This section describes what you should put into your private style file or your document if you want to use extra symbols, i.e., characters whose info in the echo area (see [Section 5.3 \[Info, page 37\]](#)) contains s.th. like `‘package.sty’` or `‘user’`. If you do not use the corresponding characters, you do not have to do anything, of course.

The TeX macros `\Box`, `\Diamond`, `\leadsto`, `\Join`, `\lhd`, `\mho`, `\rhd`, `\sqsupset`, `\sqsubset`, `\unlhd`, `\unrhd`, are defined in LaTeX package `‘latexsym.sty’`:

```
\usepackage{latexsym}
```

Note that these macros are also defined `‘amssymb.sty’`. Since the first four macros are defined differently (better) in `‘latexsym.sty’`, it does make sense to load both LaTeX packages.

The TeX macros `\boldsymbol`, `\circledast`, `\circledcirc`, `\circleddash`, `\digamma`, `\gtrapprox`, `\gtrsim`, `\lessapprox`, `\lesssim`, `\triangleq`, `\varkappa` are defined in AMS LaTeX package `‘amssymb.sty’`:

```
\usepackage{amssymb}
```

The TeX macros `\bigsqcap`, `\llbracket`, `\rrbracket`, `\llparenthesis`, `\rrparenthesis` are defined in the LaTeX package `‘stmaryrd.sty’`:

```
\usepackage{stmaryrd}
```

The TeX macros `\guilsinglleft`, `\guilsinglright`, `\dj`, `\NG`, `\ng`, `\DH`, `\DJ`, `\dh`, `\dj`, `\TH`, `\th`, `\guillemotleft`, `\guillemotright` and the ogonek characters are only defined if you use T1 font encoding:

```
\usepackage[T1]{fontenc}
```

The TeX macro `\mathmicro` for U00B5 can be defined by (see [Section 9.2.4 \[Wishlist LaTeX\]](#), [page 78](#)):

```
\let\mathmicro\mu
```

You should define the following in your LaTeX file (if you use the corresponding characters), the first can only be used with T1 font encoding.

```
\DeclareTextSymbol{\textbackslash}{T1}{92}
\newcommand{\nsubset}{\not\subset}
\newcommand{\textflorin}{\textit{f}}
\newcommand{\setB}{\{\mathord{\mathbb{B}}\}}
\newcommand{\setC}{\{\mathord{\mathbb{C}}\}}
```

```

\newcommand{\setN}{\mathord{\mathbb N}}
\newcommand{\setQ}{\mathord{\mathbb Q}}
\newcommand{\setR}{\mathord{\mathbb R}}
\newcommand{\setZ}{\mathord{\mathbb Z}}
\newcommand{\coloncolon}{\mathrel{::}}

```

The TeX macros `\textordfeminine`, `\textordmasculine`, `\textdegree`, `\textonequarter`, `\textonehalf`, `\textthreequarters`, `\mathonesuperior`, `\mathtwosuperior`, `\maththreesuperior`, `\textcopyright` are only defined when using LaTeX package ‘inputenc.sty’:

```
\usepackage[latin1]{inputenc}
```

The TeX macros `\textcent`, `\textcurrency`, `\textyen`, `\textbrokenbar`, `\textmalteseH`, `\textmalteseh` are defined as not available in LaTeX package ‘inputenc.sty’. See [Section 9.2.4 \[Wishlist LaTeX\]](#), page 78. If you use this package and want to define these commands, use `\renewcommand` (or `\def`) after, e.g.:

```

\usepackage[latin1]{inputenc}
\usepackage{wasysym} %% defines \cent, \currency, \brokenvert
\usepackage{amssymb} %% defines \yen
\renewcommand{\textcent}{\cent}
\renewcommand{\textcurrency}{\currency}
\renewcommand{\textyen}{\yen}
\renewcommand{\textbrokenbar}{\brokenvert}

```

## 6.3 Token Language “SGML entity” (sgml)

For buffers using the major mode `html-mode`, `hm--html-mode`, `html-helper-mode`, `sgml-mode` or `xml-mode`, we use token language SGML *entity* (`sgml`). This language provides the display of super-/subscripts and images. If the buffer visits a file and uses a HTML mode, X-Symbol mode is automatically turned on.

### 6.3.1 Basics of Language “SGML entity”

The standard behavior can be controlled by the following variables:

`x-symbol-sgml-modes`

`x-symbol-sgml-auto-style`

The variables known from [Section 3.3 \[Minor Mode\]](#), page 20. If the buffer uses a HTML mode, super-/subscripts and images are displayed, otherwise unique decoding (see [Section 3.2.4 \[Unique Decoding\]](#), page 18) will be used.

`x-symbol-sgml-auto-coding-alist`

Used there to automatically deduce the specific encoding of the file (see [Section 3.2.2 \[File Coding\]](#), page 16). It searches for the following string in the current buffer, including the comment:

```

<meta http-equiv="content-type"
      content="text/html; charset=encoding">

```

where *encoding* should be one of ‘iso-8859-1’, ‘iso-8859-2’, ‘iso-8859-3’, ‘iso-8859-9’, or ‘iso-8859-15’. 8bit characters are not encoded if the file if the search was successful (see [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17).

The input methods and the character info in the echo area are controlled by:

**x-symbol-sgml-header-groups-alist**

Defines the headers and their characters for the language specific Grid and Menu.

**x-symbol-sgml-extra-menu-items**

There are no special entries in the X-Symbol menu.

**x-symbol-sgml-electric-ignore**

There is no additional constraint to the ones mentioned in [Section 4.8 \[Input Method Electric\]](#), page 30.

**x-symbol-sgml-class-alist****x-symbol-sgml-class-face-alist**

Token classes (see [Section 3.6 \[Char Group\]](#), page 23) are only used to define a coloring scheme. X-Symbol uses dark orange or dark red for non-Latin-1 characters in the Grid (see [Section 4.5 \[Input Method Grid\]](#), page 27 and the character info (see [Section 5.3 \[Info\]](#), page 37), dark red for characters without defined entity names in HTML (see [Section 6.3.3 \[SGML Entity Conversion\]](#), page 49).

### 6.3.2 Super-/Subscripts and Images in HTML

The display of super- and subscripts (see [Section 5.1 \[Super and Subscripts\]](#), page 33) is controlled by:

**x-symbol-sgml-font-lock-regexp****x-symbol-sgml-font-lock-limit-regexp****x-symbol-sgml-font-lock-alist****x-symbol-sgml-font-lock-contents-regexp**

The superscript command `<sup>...</sup>` and the subscript command `<sub>...</sub>` is recognized. The contents should contain at least one character which is not a space or a `nobreakspace`.

The display of images (see [Section 5.2 \[Images\]](#), page 33) is controlled by:

**x-symbol-sgml-image-keywords**

The following commands are recognized. Extension *ext* stands for ‘gif’, ‘png’, ‘jpeg’ or ‘jpg’.

```

```

**x-symbol-sgml-master-directory****x-symbol-sgml-image-searchpath**

Relative file names (see [Section 5.2.1 \[Image Display\]](#), page 34) are relative to the directory of the current file.

**x-symbol-sgml-image-file-truename-alist**

The file name prefix ‘file:’ is ignored. For any other file name which starts with letters and then a colon, e.g., with ‘http:’ or ‘C:\’ (which is no URL anyway), the image insertion command will be skipped. By changing this variable, you could specify that the prefix ‘http://www.fmi.uni-passau.de/~wedler/’ corresponds to ‘~/public\_html/’.

**x-symbol-sgml-image-cached-dirs**

The file name in the image command should not have a directory part or the directory part should be ‘images/’ or ‘pictures/’ if the image should be cached in the memory cache.

### 6.3.3 The Conversion of SGML Entities

Most character entities of HTML-4.0 are supported, except the following: uppercase Greek which look like uppercase Latin, “markup-significant and internationalization” characters, and some quotes. See <http://www.w3.org/TR/REC-html40/sgml/entities.html>.

By default, we encode to entity references like `&amp;`, and decode from both entity references and character references like `&#38;`. For Latin-N characters without defined entity names in HTML (e.g. `scedilla`), we can only use character references.

Do not expect Netscape before v6 to display non-Latin-1 characters correctly (this might work by specifying the charset UTF-8 and using character references).

You might want change the conversion between characters and tokens in language `sgml` by changing:

`x-symbol-sgml-token-list`

A symbol, which defines whether to use entity references, character references, or entity references for Latin-1 characters and character references for others.

`x-symbol-sgml-user-table`

It is probably not a good idea to change the defined tokens (except via the variable above), but you might want to add some definitions:

```
(setq x-symbol-sgml-user-table '((circ () 999 "&bcomp;")))
```

## 6.4 Token Language “BibTeX macro” (`bib`)

For buffers using the major mode `bibtex-mode`, we use token language *BibTeX macro* (`bib`). This language does not provide the display of super-/subscripts and images. If the buffer visits a file, X-Symbol mode is automatically turned on. It is controlled by:

`x-symbol-bib-modes`

`x-symbol-bib-auto-style`

The variables known from [Section 3.3 \[Minor Mode\]](#), page 20. There is no automatic deduction of the file encoding, 8bit characters are usually encoded, and there is usually no unique decoding. See [Section 3.2 \[Conversion\]](#), page 15.

The major difference between this language and the token language `tex` is that the tokens for text-mode characters are most likely enclosed by braces. This has some problems (see [Section 6.2.3 \[TeX Macro Problems\]](#), page 43), but is required by the program `bibtex`.

The input methods and most features except super-/subscripts and images work like in token language `tex` (see [Section 6.2 \[TeX Macro\]](#), page 41):

`x-symbol-bib-header-groups-alist`

`x-symbol-bib-electric-ignore`

`x-symbol-bib-class-alist`

`x-symbol-bib-class-face-alist`

Like in [Section 6.2.2 \[TeX Macro Features\]](#), page 42.

`x-symbol-bib-extra-menu-items`

There are no special entries in the X-Symbol menu.

You might want change the conversion between characters and tokens in language `bib` by changing:

`x-symbol-bib-user-table`

`x-symbol-tex-user-table`

Use the former for `bib`-only changes, the latter also influences the conversion with token language `tex`.

## 6.5 Token Language “ $\text{\TeXinfo}$ command” (`texi`)

For buffers using the major mode `texinfo-mode`, we use token language  *$\text{\TeXinfo}$  command* (`texi`). This language does not provide the display of super-/subscripts and images. If the buffer visits a file, X-Symbol mode is automatically turned on. It is controlled by:

`x-symbol-texi-modes`

`x-symbol-texi-auto-style`

The variables known from [Section 3.3 \[Minor Mode\]](#), page 20. There is no automatic deduction of the file encoding, 8bit characters are usually encoded, and there is usually no unique decoding. See [Section 3.2 \[Conversion\]](#), page 15.

With `x-symbol-8bits` having value `nil` (the default), it might still happen that the saved file contains 8bit characters, since token language `texi` does not define tokens for all characters in the Latin charsets supported by X-Symbol. See [Section 3.2.3 \[Controlling 8bit Coding\]](#), page 17.

With `x-symbol-unique` having value `nil` (the default), we have unique decoding anyway, since token language `texi` does only define one token per character, i.e., the value is not important if `x-symbol-8bits` is `nil`. See [Section 3.2.4 \[Unique Decoding\]](#), page 18.

The input methods and the character info in the echo area are controlled by:

`x-symbol-texi-header-groups-alist`

Defines the headers and their characters for the language specific Grid and Menu.

`x-symbol-texi-extra-menu-items`

There are no special entries in the X-Symbol menu.

`x-symbol-texi-electric-ignore`

There is no additional constraint to the ones mentioned in [Section 4.8 \[Input Method Electric\]](#), page 30.

`x-symbol-texi-class-alist`

`x-symbol-texi-class-face-alist`

Only a few token classes (see [Section 3.6 \[Char Group\]](#), page 23) are defined, the most interesting induces the character info (see [Section 5.3 \[Info\]](#), page 37) to display ‘not as code’ for `@minus{}` (`@minus{}` should not be used inside `@code` and `@example`). No coloring scheme is defined.

At least with `makeinfo-4.0`, you do not get accented characters in the info file for the corresponding  $\text{\TeXinfo}$  commands in the ‘`.texi`’ file, the HTML output might contain illegal “SGML entities” like `&140;`.

At least with `texi2html-1.62`, you see accented characters in the HTML output for the corresponding  $\text{\TeXinfo}$  commands in the ‘`.texi`’ file, but the output might also contain illegal “SGML entities” like `&140;`.

You might want change the conversion between characters and tokens in language `texi` by changing:

`x-symbol-texi-user-table`

Extra entries for the conversion.

## 6.6 Languages Defined in Other Emacs Packages

It is no problem for other Emacs packages to define their own token language (see [Section 7.4 \[Extending X-Symbol\]](#), page 55).

I know of the following package—please check its manual for details.

- Package **ProofGeneral** defines token language “Isabelle symbol”.

## 7 X-Symbol Internals

This section is outdated, it currently describes Version 3.4.2 of X-Symbol.

Package X-Symbol is distributed in two ways. End-users should use the *binary package* which contains pre-compiled files. X-Symbol developers should use the *source package* which contains some additional files.

### 7.1 Internal Representation of X-Symbol Characters

As mentioned in [Section 6.1 \[Pseudo Language\]](#), [page 41](#), most functions do not operate on X-Symbol characters directly, they use “x-symbol charsyms”. These charsyms have a symbol property `x-symbol-cstring` which points to a string, called *cstring*, containing the X-Symbol character.

- Under Emacs and XEmacs/Mule, the string only contains the character which is a normal Mule character created by `make-char`.
- Under XEmacs/no-Mule, the string only contains the 8bit character if the X-Symbol character is a 8bit character according to `x-symbol-default-coding` (see [Section 3.2.1 \[Default Coding\]](#), [page 15](#)). Otherwise, the string contains of a *leading character* (with range ‘\200’ to ‘\237’) and an *octet*. Package `font-lock` is used to display them correctly as X-Symbol characters (see [Section 8.4.3 \[FAQ Strange Chars\]](#), [page 65](#)). E.g., with ‘\251’ is copyright, we get

```
(get 'Idotaccent 'x-symbol-cstring)
⇒ "\235\251"
```

If the character is also a 8bit character in some encoding (see [Section 3.2.2 \[File Coding\]](#), [page 16](#)), the charsym also has the symbol property `x-symbol-file-cstrings` for the representation in the file and property `x-symbol-buffer-cstrings` to recognize character aliases (see [Section 3.2.7 \[Char Aliases\]](#), [page 20](#)). E.g., under XEmacs/no-Mule, with ‘\335’ is Yacute, ‘\251’ is copyright, we get

```
(get 'Idotaccent 'x-symbol-file-cstrings)
⇒ (iso-8859-9 "\335" iso-8859-3 "\251")
(get 'Idotaccent 'x-symbol-buffer-cstrings)
⇒ (iso-8859-9 "\234\335" iso-8859-3 "\235\251")
```

The values are plists (see [section “Property Lists” in XEmacs Lisp Reference Manual](#)) mapping the file coding to the strings in the file or the buffer, respectively.

After token languages have been initialized, the charsym also has the symbol properties `x-symbol-tokens` (see [Section 3.1 \[Token Language\]](#), [page 15](#)) and `x-symbol-classes` (see [Section 3.6 \[Char Group\]](#), [page 23](#)):

```
(get 'Idotaccent 'x-symbol-tokens)
⇒ (sgml "&#304;" tex "{\\.\I}")
(get 'Idotaccent 'x-symbol-classes)
⇒ (sgml (non-l1) tex (text aletter))
```

### 7.2 Defining X-Symbol Charsets

An X-Symbol charset, called *cset* in the code and the docstrings, handles one font used by package X-Symbol. Each cset must use the same char registry—encoding as the corresponding variables for the fonts (see [Section 2.9 \[Installing Fonts Lisp\]](#), [page 12](#)).



You have to tell X-Symbol, how to define Mule charsets with Emacs or XEmacs/Mule and which leading character to use with XEmacs/no-Mule. As an example, we use the definition of the Adobe symbol font.

```
(defvar x-symbol-xsymb0-cset
  '(((("adobe-fontspecific") ?\233 -3600)
    (xsymb0-left "X-Symbol characters 0, left" 94 ?:) .
    (xsymb0-right "X-Symbol characters 0, right" 94 ?\;))))
```

Mule charsets (see [section “Charsets” in XEmacs Lisp Reference Manual](#)) may be used for 94 or 96 characters (this example: 94, only charset with dimension 1 can be defined with X-Symbol). Thus, if your font provides more characters, you are likely to use both the left and the right half of the font to define two Mule charsets. For both of them, you have to define a unique, free final character/byte of the standard ISO 2022 escape sequence designating the charset (this example: ‘:’ and ‘;’). The remaining free (reserved by Emacs for users) are ‘>’ and ‘?’, the latter is already used in XEmacs.

For XEmacs/no-Mule, you have to define the leading character (this example: ‘\233’).

```
x-symbol-latin1-cset
x-symbol-latin2-cset
x-symbol-latin3-cset
x-symbol-latin5-cset
```

Cset definitions only using the upper halves of the fonts where the corresponding Mule charsets are known and which define characters which are considered 8bit characters in the corresponding encoding, see [Section 3.2.2 \[File Coding\]](#), page 16.

```
x-symbol-xsymb0-cset
x-symbol-xsymb1-cset
```

Cset definitions using both halves of the fonts where no corresponding Mule charset are yet known.

## 7.3 Defining Input Methods

This is probably the hardest section in this manual. . . .

### 7.3.1 Defining Input Methods: Objectives

Input methods should be intuitive. This requires consistency:

- Characters should be found under the same header in the Grid and in the Menu.
- If one character can be modified or rotated to another character (see [Section 4.7 \[Input Method Context\]](#), page 29), both should stand near to each other in the Grid. E.g., since `arrowsouthwest` rotates to `arrowdown`, they stand next to each other.
- The key binding should be similar to the context of input method Context. If two characters are defined to have the same context, they should have the same key prefix and the suffix should be a number which increases with the “modify-to” behavior. E.g., `reflexsubset` with key binding `C-= < _ 2` modifies to `reflexsqssubset` with key binding `C-= < _ 3`.
- Consistent definition of “modify-to” and “rotate-to”: if A can be modified to B and rotated to C and C can be modified to D, B can be rotated to D in most cases.
- It should be possible to load character definitions later on, e.g., when new token languages get initialized.
  - Existing key bindings should not be overwritten. If some of them have to change, it should be done in a uniform way (solution: key suffix ‘1’).



- Also, modifying or rotating a new character to/from old ones should be possible without changing the input definitions of the old characters.

Observation: It is impossible, especially with the possibility to load character definitions later on, to define the input methods directly, i.e., by something like `define-key`. The solution is an indirect definitions with “character descriptions”.

### 7.3.2 X-Symbol Character Descriptions: Example

As an example for “character descriptions”, look at the definition of `longarrowright` in `x-symbol-xsyml-table` (‘95’ is the encoding in the font and not of interest here). Some terms are defined in the next section:

```
(longarrowright 95
  (arrow) (size big . arrowright) nil ("->" t "-->") (emdash))
```

With this definition, package X-Symbol automatically defines:

- Key bindings `C-- - - >` and `C-- - - > 2`, the latter has suffix 2, because `C-- - - >` is also “wanted” by `arrowright` which now has the key binding `C-- - - > 1` (the “score” of `longarrowright` is higher, due to ‘size big’). See [Section 4.6 \[Input Method Keyboard\]](#), page 28.
- `arrowright` modifies to `longarrowright`, which modifies to `arrowright`. See [Section 4.7 \[Input Method Context\]](#), page 29.
- `longarrowleft` rotates to `longarrowright`, which rotates to `longarrowboth` (which rotates to `longarrowleft`). (The “rotate aspects” are inherited from `arrowright`.) See [Section 4.7 \[Input Method Context\]](#), page 29.
- The following contexts can be modified to `longarrowright`: ‘-->’ or `minus1 / emdash / macron / emdash / hyphen` and ‘->’ (since all define context ‘-’) and `emdash` and ‘>’ (since `emdash` defines context ‘--’). ‘->’ is used for `arrowright`, which has a lower score, see above. See [Section 4.7 \[Input Method Context\]](#), page 29.
- Input method Electric will change context ‘-->’ (is tagged with `t` in the definition) to `longarrowright`, also `emdash` and ‘>’ (only theoretically, since input method Electric will produce `emdash` only in T<sub>E</sub>X’s text mode, and `longarrowright` only in T<sub>E</sub>X’s math mode). See [Section 4.8 \[Input Method Electric\]](#), page 30.
- The character will appear in the Grid under the header ‘Arrow’. You will probably recognize that the placement is based on the modify-to and rotate-to behavior above. See [Section 4.5 \[Input Method Grid\]](#), page 27.
- The character will appear in the Menu under one of the headers ‘Arrow n’. The submenus are sorted alphabetically. See [Section 4.4 \[Input Method Menu\]](#), page 26.

Consider that this character would be missing in package X-Symbol and you want to define your own character (in your own font). With the current scheme, the one line above is enough! Have fun defining all the consequences directly instead. . . .

### 7.3.3 Defining Input Methods by Character Descriptions

Characters are defined with *character descriptions* which consist of different *aspects* and *contexts*, which can also be inherited from a *parent* character. All characters which are connected with parents, form a *component*. Aspects and contexts are used to determine the modify-to and rotate-to chain for characters, the contexts for input method Context and Electric, the key bindings, and the position in the Menu and the Grid.

If you want to check the component, scores, etc of a specific character, look at the symbol property (e.g., with *M-x hyper-apropos-get-doc*) of the corresponding charsym, e.g., *arrowright*. See also the docstrings of *x-symbol-init-cset* and *x-symbol-init-input*.

Remember, all characters which are connected with parents, form a component. *Contexts* are the contexts of input method Context (see [Section 4.7 \[Input Method Context\]](#), page 29). If a table entry of a charsym does not define its own contexts, they are the same as the contexts of the charsym in an earlier position in the modify chain (see below), or the contexts of the first charsym with defined contexts in the modify chain. The *modify context* of a charsym is the first context.

#### *x-symbol-rotate-aspects-alist*

Characters in the same component whose aspects only differ by their **direction** (*east*,...), a key in this alist, are circularly connected by “rotate-to”. The sequence in the *rotate chain* is determined by *rotate scores* depending on the values in the *rotate aspects*. Charsyms with the same “rotate-aspects” are not connected (charsyms with the smallest modify scores are preferred).

```
(get 'longarrowright 'x-symbol-rotate-aspects)
⇒ (-1500 direction east)
```

#### *x-symbol-modify-aspects-alist*

Characters in the same components whose aspects only differ by their **size** (*big*,...), **shape** (*round*, *square*...) and/or **shift** (*up*, *down*,...), keys in this alist, are circularly connected by “modify-to”, if all their modify contexts are used exclusively, i.e., no other modify chain uses any of them. The sequence in the *modify chain* is determined by *modify scores* depending on the values in the *modify aspects*, the charsym score defined in the definition tables and the score of the whole cset (see [Section 7.2 \[Defining Charsets\]](#), page 51).

```
(get 'longarrowright 'x-symbol-score)
⇒ -3500
(get 'longarrowright 'x-symbol-modify-aspects)
⇒ (1500 shift nil shape nil size big)
```

Otherwise, the “modify chain” is divided into modify subchains, which are those charsyms sharing the same modify context. All modify subchains using the same modify context, build a *horizontal chain* whose charsyms are circularly connected by “modify-to”.

We build a *key chain* for all contexts (not just modify contexts), consisting of all charsyms (sorted according to modify scores) having the context. Input method Context modifies the context to the first charsym in the key chain.

#### *x-symbol-key-suffix-string*

If there is only one charsym in the key chain, *C=* plus the context inserts the charsym. Otherwise, we determine a suffix for each charsym in the key chain by its index and this string. *C=* plus the context plus the suffix inserts the charsym.

### 7.3.4 Defining Input Methods: Example

An example:	Modify Score	Modify Aspect	Rotate Score	Rotate Aspect	Modify Context	Other Contexts
charsym 1w	150	nil	100	west	'a'	'c'
charsym 2w	200	nil	100	west	'b'	-
charsym 3w	350	big	100	west	('b')	(-)
charsym 1e	100	nil	200	east	('a')	('b')
charsym 2e	250	big	200	east	'a'	'b'

charsym 3e	300	big	200	east	'a'	-
charsym 1n	100	nil	300	north	'd'	'c'
charsym 2n	200	big	300	north	'c'	-

Assuming that all charsyms form one component, we have:

Rotate chains:	(1w,2w)-1e-1n and 3w-(2e,3e)-2n.
Modify chains:	1w-2w-3w and 1e-2w-3w and 1n-2n.
Horizontal chains:	1e-1w-2e-3e (for modify context 'a')
	2w-3w (for modify context 'b')
Key chains:	1e-1w-2e-3e (for context 'a')
	1e-2w-2e-3w (for context 'b')
	1n-1w-2n (for context 'c')
	1n (for context 'd')

That makes the following bindings:

```

Rotate-to: 1w->1e, 2w->1e, 1e->1n, 1n->1w
           3w->2e, 2e->2n, 3e->2n, 2n->3w
Modify-to: 1e->1w, 1w->2e, 2e->3e, 3e->1e (horizontal chain)
           2w->3w, 3w->2w (horizontal chain)
           1n->2n, 2n->1n (modify chain with exclusive modify contexts)
CONTEXTS: 'a'->1e, 'b'->1e, 'c'->1n, 'd'->1n
KEY:      'a1'=1e, 'a2'=1w, 'a3'=2e, 'a4'=3e, 'b1'=1e, ..., 'd'=1n

```

### 7.3.5 Customizing Input Methods

When defining contexts for characters, you should try to use default contexts to make them and key bindings as consistent as possible. E.g., package X-Symbol only defines explicit contexts for 186 of the 437 characters.

#### x-symbol-group-input-alist

Defines default scores and bindings for characters of a group (see [Section 3.6 \[Char Group\]](#), page 23). E.g., the definition (in x-symbol-latin1-table)

```
(aacute 225 (acute "a" Aacute))
```

defines `aacute` without any explicit contexts, but having the group `acute` and the subgroup `'a'`. The default input for the group is defined by the following element in this variable:

```
(acute 0 "%s'" t "'%s")
```

That means: 0 is added to the normal “modify-score” of the character. `'%s'` and `'%s'` with `%s` substituted by the subgroup, i.e., `'a'` and `'a'`, are the contexts for `aacute`. The context `'a'` is also used for input method Electric since it is prefixed by `t`.

#### x-symbol-key-min-length

It is quite unlikely that a one-character context is not the prefix of another context, at least when loading additional font definitions. In order not to have to change key bindings `C-= key` to `C-= key 1`, it is required that the length of the key binding without `C-=` is at least 2.

## 7.4 Extending Package X-Symbol

In this section, you are told what to consider and what to do when extending package X-Symbol with new characters and new token languages. If you only want to define a token language using existing characters, you only have to read the last section.

### 7.4.1 Extending X-Symbol with New Fonts

If you add a new token language to package X-Symbol which should represent tokens by characters which are not yet defined by package X-Symbol, you have to add a new font to package X-Symbol, first.

When adding new fonts to package X-Symbol, consider that X-Symbol has to run under Emacs, XEmacs/Mule and XEmacs/no-Mule.

Running under Emacs and XEmacs/Mule requires that you cannot use all encodings in a font for characters: you should probably only use encodings 33 to 126 and 160 to 255. You should also use a unique pair of charset properties ‘CHARSET\_REGISTRY’ and ‘CHARSET\_ENCODING’.

Running under XEmacs/no-Mule can lead to problems when major modes do not check whether the previous character is an escape character (in our case, a leading character, see [Section 7.1 \[Char Representation\]](#), page 51) when looking at a character. Thus, you should probably not use encodings which represent characters in your default font with a special syntax.

- In general, escape sequences use the digits of the current font. Thus, you should probably define the encodings 48 to 57 as digits ‘0’ to ‘9’.
- In LaTeX buffers, characters in ‘\$%\{’ have a special syntax. Thus, you should probably not use encodings 36, 37, 92, 123 and 125 for characters which could also be useful with token languages `tex` and `utex`.
- In HTML buffers, characters in ‘&<>’ have a special syntax. Thus, you should probably not use encodings 38, 60 and 62 for characters which could also be useful with token language `sgml`.

You have to tell package X-Symbol which fonts to use for the normal text, subscripts and superscripts. See [Section 2.9 \[Installing Fonts Lisp\]](#), page 12.

You have to tell X-Symbol, how to define Mule charsets with Emacs and XEmacs/Mule and which leading character to use with XEmacs/no-Mule. See [Section 7.2 \[Defining Charsets\]](#), page 51.

### 7.4.2 Guidelines for Input Definitions

Read section [Section 7.3 \[Defining Input Methods\]](#), page 52. Look at the tables in ‘`x-symbol.el`’. Here are some guidelines of how to define the input methods for new characters:

1. Define reasonable character groups for new characters, see [Section 3.6 \[Char Group\]](#), page 23. E.g., if you add the IPA font for phonetic characters, you are likely to define at least one additional charset group. If you do not know whether to use one or two groups for a set of characters, use two.
2. Define under which Grid/Menu header the character of the new character group should appear. You may also want to add additional headers for these characters. See [Section 3.6 \[Char Group\]](#), page 23.
3. If reasonable, define default contexts for characters of a group, see [Section 7.3.5 \[Customizing Input Methods\]](#), page 55.
4. For the other characters, define contexts by Ascii sequences which look similar to the character.
5. Form a component for a set of characters which are strongly related to each other. In most cases, characters of a component are in the same group but not vice versa. E.g., the simple arrows already defined by package X-Symbol form one component. You form a component of characters by specifying parents in their definition, see [Section 7.3.3 \[Char Descriptions\]](#), page 53.

6. Use aspects to describe the new characters. Add new aspects to `x-symbol-modify-aspects-alist` and `x-symbol-rotate-aspects-alist` if necessary (see [Section 7.3.3 \[Char Descriptions\]](#), page 53).
7. Finish the definition of your font file (see [Section 7.4.3 \[Font Definition File\]](#), page 57), load it with `M-x load-file`, and initialize the input methods, e.g., by invoking the grid (`M-x x-symbol-grid`).
8. If there are no errors, you are likely to get warnings about equal modify scores. In this case, the sequence of characters in the modify-to chain is random, so are the numerical suffixes of key bindings.
  - a. Define a base score for the whole X-Symbol charset (“cset score”) which should be a positive number in order not to change the key bindings of previously defined X-Symbol characters.
  - b. Define reasonable scores for newly defined aspects and character groups.
  - c. Finally, fine-tune your definitions by charsym scores in the tables. This should be necessary only for a few characters.

### 7.4.3 Emacs Lisp File Defining a New Font

Now put all things together in a separate font definition file. You should not put it in a language definition file.

Here is a tiny example using only the lower half of the font:

```
(provide 'x-symbol-myfont)
(defvar x-symbol-myfont-fonts
  '(("x-symb-myfont-medium-r-normal--14-140-75-75-p-85-x symb-myfont")
    ("-x symb-myfont_sub-medium-r-normal--12-120-75-75-p-74-x symb-myfont")
    ("-x symb-myfont_sup-medium-r-normal--12-120-75-75-p-74-x symb-myfont")))
(defvar x-symbol-myfont-cset
  '(("x symb-myfont") ?\200 1000)
  (myfont-left "My font characters, left" 94 63) . nil))
(defvar x-symbol-myfont-table
  '((longarrownortheast 33 (arrow) (size big . arrownortheast))
    (koerper 34 (setsymbol "K"))
    (circleS 35 (symbol "S") nil nil "S0")))
(x-symbol-init-cset x-symbol-myfont-cset x-symbol-myfont-fonts
  x-symbol-myfont-table)
```

Due to an XEmacs bug with char syntax `inherit`, you should also add the following line to files `'x-symbol-xmas20.el'` and `'x-symbol-xmas21.el'`:

```
(modify-syntax-entry ?\200 "\\\" (standard-syntax-table))
```

### 7.4.4 Emacs Lisp File Extending a Token Language

If you want to use the new font to extend an existing token language, define a new token language which inherits most variables from the “parent language”. E.g., token language `utex` inherits most variables from `tex`, see `'x-symbol-utex.el'`.

A language must define variables for all language aspects, see [Section 7.7 \[Language Internals\]](#), page 61. Our example defines a language `mytex` using the additional characters from [Section 7.4.3 \[Font Definition File\]](#), page 57.

First, you have to register the language in a startup file:

```
(defvar x-symbol-mytex-name "My TeX macro")
(defvar x-symbol-mytex-modes nil)
(x-symbol-register-language 'mytex 'x-symbol-mytex x-symbol-mytex-modes)
```

The language definition file should look like (leaving out most parts which are similar to the ones in 'x-symbol-utex.el'):

```
(provide 'x-symbol-mytex)
(require 'x-symbol-tex)
(defvar x-symbol-mytex-required-fonts '(x-symbol-myfont))
(put 'mytex 'x-symbol-font-lock-keywords 'x-symbol-tex-font-lock-keywords)
(defvar x-symbol-mytex-user-table nil)
(defvar x-symbol-mytex-myfont-table
  '((longarrownortheast (math arrow user) "\\longnortheastarrow")
    (koerper (math letter user) "\\setK")
    (circleS (math ordinary amssymb) "\\circledS")))
(defvar x-symbol-mytex-table
  (append x-symbol-mytex-user-table
    '(nil)
    x-symbol-mytex-myfont-table
    x-symbol-tex-table))
```

It is important that you do not define a variable for the language access `x-symbol-font-lock-keywords`, but rather use the variable of the parent language directly, see [Section 7.7 \[Language Internals\]](#), page 61.

During the testing phase, you should probably leave out the `'(nil)'` which prevents warnings about redefinitions for the following elements.

### 7.4.5 Emacs Lisp File Defining a New Token Language

You might also want to define a new token language not based on another language.

As an example, consider a token language “My Unicode” (`myuc`) for buffers with major mode `myuc-mode`. Thus, we register the language by:

```
(defvar x-symbol-myuc-name "My Unicode")
(defvar x-symbol-myuc-modes '(myuc-mode))
(x-symbol-register-language 'myuc 'x-symbol-myuc x-symbol-myuc-modes)
```

Each token if language `myuc` consists of `#` plus the hexadecimal representation of the Unicode with hexadecimal values where the case of digits is not important and the preferred case is upcase. A single `#` is represented by the token `##`. In order to be more flexible, we want to define the tokens by their decimal value in the table. There are no subscript and no images. The code below (`'x-symbol-myuc.el`) is included in the source distribution of package X-Symbol.

```
(provide 'x-symbol-myuc)
(defvar x-symbol-myuc-required-fonts nil)
(defvar x-symbol-myuc-modeline-name "myuc")
(defvar x-symbol-myuc-class-alist
  '((VALID "My Unicode" (x-symbol-info-face))
    (INVALID "no My Unicode" (red x-symbol-info-face))))
(defvar x-symbol-myuc-font-lock-keywords nil)
(defvar x-symbol-myuc-image-keywords nil)
...
(defvar x-symbol-myuc-case-insensitive 'upcase)
(defvar x-symbol-myuc-token-shape '(?# "[0-9A-Fa-f]+\\\'" . "[0-9A-Fa-f]"))
```



```

(defvar x-symbol-myuc-exec-specs '(nil (nil . "[0-9A-Fa-f]+")))
(defvar x-symbol-myuc-input-token-ignore nil)

(defun x-symbol-myuc-default-token-list (tokens)
  (list (format "%X" (car tokens))))
(defvar x-symbol-myuc-token-list 'x-symbol-myuc-default-token-list)
(defvar x-symbol-myuc-user-table nil)
(defvar x-symbol-myuc-xsymb0-table
  '((alpha () 945) (beta () 946)))
(defvar x-symbol-myuc-table
  (append x-symbol-myuc-user-table x-symbol-myuc-xsymb0-table))
...
```

## 7.5 Various Internals

### 7.5.1 Tagging Insert Commands for Token and Electric

Input methods Token (see [Section 4.2 \[Input Method Token\], page 26](#)) and Electric (see [Section 4.8 \[Input Method Electric\], page 30](#)) stop their auto replacement if you use a command which is not an insert command.

```

self-insert-command
newline
newline-and-indent
reindent-then-newline-and-indent
tex-insert-quote
TeX-insert-quote
TeX-insert-punctuation
TeX-insert-dollar
sgml-close-angle
sgml-slash
```

These commands and commands aliased to these are recognized as input commands by having a non-nil value of its symbol property `x-symbol-input`.

### 7.5.2 Avoiding Hide/Show-Invisible Flickering

Starting a command makes a previously revealed super- or subscript command (see [Section 5.1 \[Super and Subscripts\], page 33](#)) invisible again. Repeatedly invoking commands which moves the point just by a small amount can lead to some flickering.

```

forward-char
forward-char-command
backward-char
backward-char-command
```

If the point position after the execution of these commands is still “at” the super- or subscript command, the command won’t be made invisible at the first place. Each of these four commands have a function (1+ and 1-) as the value of its symbol property `x-symbol-point-function` which returns the position “after” when called with the position “before”.



## 7.6 Design Alternatives

This section describes potential design alternatives and why they were not used.

### 7.6.1 Alternative Token Representations

Package X-Symbol represents tokens in the file by characters in the buffer. This requires an automatic conversion when visiting a file or saving a buffer, see [Section 3.2 \[Conversion\]](#), page 15.

Another possibility would be to use the tokens directly in the buffer and just display them differently. You would need no conversion and you could copy the text easily to a message buffer. This could be done by a special face and an additional font-lock keyword for every token. The disadvantages make this approach unfeasible:

- The editing commands would work on the tokens which are invisible for the user.
- Extremely resource and startup-time consuming. If as many characters should be supported as done by package X-Symbol, including superscripts and subscripts, more than 2000 faces with display tables would have to be defined even without considering char aliases!
- Time consuming. More than 2000 entries in you font-lock keywords would slow down the fontification considerably, which would be too much even when using `lazy-shot`!

Another possibility would be to adapt T<sub>E</sub>X to the representations of the corresponding characters in Emacs' buffer. Again, you would need no conversion. The disadvantages make this approach too restrictive:

- You cannot adopt SGML to this approach.
- You cannot read normal LaT<sub>E</sub>X files directly, you do not write normal LaT<sub>E</sub>X files.
- You would have different T<sub>E</sub>X versions: one for X-Symbol with Emacs and XEmacs/Mule, one with XEmacs/no-Mule.
- If you are not an extremely good T<sub>E</sub>X hacker, it would be impossible to adopt this approach to support more than 256 characters.

A third alternative would be very similar to the method used in this package. There would be just a slight difference when running under XEmacs/no-Mule: the internal representation of a character is always just one character, but we would also provide font properties for characters not of your default font. The disadvantages make this approach too unsafe:

- Problems with current search/replace commands.
- Problems with the current version of `font-lock` (it should *never* overwrite the font property for this character, even if the character matches some *match* in `font-lock-keywords` and `overwrite` is non-`nil`). This gets even more difficult with superscripts/subscripts.
- Unless you can provide a syntax table for faces (you cannot), characters in different faces with the same encoding are in the same syntax class, which is irritating: e.g., `\leftrightharpoon` and `\approx` would be delimiters.

### 7.6.2 Alternative Ways to Turn on X-Symbol Globally

This package hooks itself into `hack-local-variables-hook` which makes the installation very simple.

Another possibility would be to use the major-mode hooks which is the normal way how to turn on a minor mode. The disadvantages are:

- The installation is more complicated.

- Local variables in files are not yet processed (this was the main reason not to do it this way).

Another possibility would be to hook X-Symbol into `find-file-hooks`, as it is done in old versions of package X-Symbol. It would be as easy as the current approach but we would have to be careful with sequence of functions in `find-file-hooks`, especially with the function hooked in by `font-lock`.

### 7.6.3 Alternative Auto Conversion Methods

Without package `crypt`, this package automatically decodes tokens when turning on the minor mode (in `hack-local-variables-hook`, see [Section 7.6.2 \[Alt Global Mode\]](#), page 60) or in `after-insert-file-functions`. This package automatically encodes characters in `write-region-annotate-functions`. The disadvantage is that the possibility to change buffers in `write-region-annotate-functions` is not official (see [Section 9.2.3 \[Wishlist Emacs\]](#), page 78), i.e., not mentioned in the docstring (only mentioned for corresponding encode-functions of package `format` which use a similar loop in the C code).

With package `crypt`, this package automatically decodes tokens when turning on the minor mode. This package automatically encodes characters in `write-file-hooks`. The disadvantage is that the encoding is slower (use `jka-compr` instead `crypt`) and the problem with `vc-next-action` (see [Section 8.2 \[Spurious Encodings\]](#), page 63).

Without package `crypt`, Version 2.6 of this package automatically encoded characters in `write-file-data-hooks`. The advantage was that changing buffers there is official, the disadvantage is that it is also more complicated.

A totally different method would be to use package `format`. Unfortunately, this is not really possible, since a `regexp` in `format-alist` is much too weak, i.e., X-Symbol's decoding does not change any file headers which would represent the file format. In XEmacs, this package also fails to work properly with `jka-compr` and `crypt`.

## 7.7 Language Internals

In order to use a token language or accessing one of the language dependent values, the following conditions must be met:

- The language must be *registered*. This makes it possible to select the language in the menus. It also prevents to load a potentially dangerous file when a file specifies a buffer-local value of `x-symbol-language`.

`x-symbol-register-language`

Registering a language includes stating the name of the feature (i.e., a file) which provides the language. The name of the language must have been already defined.

- The file providing the language must have been *loaded*. This will be done automatically when the language is initialized. Customizing X-Symbol will also load the language files.
- The language must be *initialized*. This will be done automatically if the language is used. This loads the language file and fails if the language has not been registered. If some minor language information is needed, e.g., in the highlight menu of the Grid (see [Section 4.5 \[Input Method Grid\]](#), page 27), you should initialize the language explicitly, e.g., by the following command:

`M-x x-symbol-init-language-interactive`

Initialized a token language if it is not already initialized.

Language dependent values are accessed by language accesses:

#### `x-symbol-language-value`

Returns the language depending value. Also initializes language if necessary. E.g., we get the name of a language by the language access `x-symbol-name`. With a simplified expansion, we get

```
(x-symbol-language-value 'x-symbol-name 'tex)
  ↳ (symbol-value (get 'tex 'x-symbol-name))
  ⇒ (symbol-value 'x-symbol-tex-name)
  ⇒ "TeX macro"
```

#### `x-symbol-language-access-alist`

List of all language accesses. A token language *must* define all variables accessed by language accesses. A *language access* is a property of the language symbol, its value is the symbol naming a variable whose value is used.

If the language is a derived language, e.g., like language `utex`, the language access `x-symbol-font-lock-keywords`, should point directly to the variable of the parent language (here `tex`), see file `'x-symbol-utex.el'`.

## 7.8 Miscellaneous Internals

TODO. This is currently just a collection of unrelated stuff.

Characters might also define a *subgroup* which is a string defining some order on characters in the same group (see [Section 3.6 \[Char Group\]](#), page 23) and is also used for default contexts/bindings (see [Section 7.3.5 \[Customizing Input Methods\]](#), page 55).

#### `x-symbol-group-syntax-alist`

Lists all valid character groups. Under Emacs and XEmacs/Mule, this list also determines the syntax of characters.

The character group could probably also be used to define character categories if they are implemented in XEmacs.

## 8 Problems, Troubleshooting

This section is based on a successful installation of package X-Symbol. See [Section 2.11 \[Checking Installation\]](#), page 13.

### 8.1 Problems under XEmacs/no-Mule

If you use package X-Symbol under XEmacs/no-Mule, there are some annoyances which result from the fact that additional “X-Symbol characters” are represented by two characters internally. Package X-Symbol just provides a kind of “*poor* man’s Mule”, see [Section 3.4 \[Poor Mans Mule\]](#), page 22. This means: I have provided workarounds for the most annoying ones, but some remain (and will remain: I am not going to provide workarounds for these):

- If `font-lock` is not prepared to display these two-character sequences, i.e., if you installation is incomplete (see [Section 3.5 \[Role of font-lock\]](#), page 22), they look like ‘\233a’ instead alpha.
- Commands which add more than one entry to the `buffer-undo-list` and involve X-Symbol characters might lead to strange results, e.g. `C-t` (`transpose-chars`) with point between character alpha and ‘b’, leads to beta‘a’. Simple deletion and insertion works OK, though.
- Selecting or inserting a rectangle with X-Symbol characters on the left or right margin might not work properly.
- Be careful with `M-%` (`query-replace`): the first character of *from-string* can probably match the second of the two “internal” characters of an X-Symbol character.
- If you use `C-x ’` (`expand-abbrev`) without `M-’` (`abbrev-prefix-mark`) and the last word before point starts directly after a X-Symbol character, `C-x ’` could behave strange:
  - If `words-include-escapes` is `t`, there will be no expansion.
  - If `words-include-escapes` is `nil`, the second “internal” character could be the first character of the last word before point which is going to be replaced by the abbrev mechanism.
- If the character under point is a X-Symbol character, you will not see the cursor if you exit a command with an error or with quit (`C-g`). Unfortunately, XEmacs (as opposed to Emacs) does not run the hooks in `post-command-hook` in these cases. Solution: move point right (`C-f`).
- If you provide prefix arguments to commands, they are likely to consider just “internal” characters. E.g., `C-u 2 C-f` before alpha behaves like `C-f`.
- Column position considers “internal” characters, e.g., `C-n` might jump to an unexpected position (well, typically just one character left/right from the expected position, if at all).
- Auto-filling also considers “internal” characters, i.e., might break the line too early.
- There are no syntax definitions for the new characters, e.g., `M-C-f` before `floorleft` does not move to the closing `floorright`.
- In some cases, e.g., when using the minibuffer for input via `M-%` or `C-s`, the internal representation of X-Symbol characters (see [Section 7.1 \[Char Representation\]](#), page 51) are displayed directly (see [Section 8.4.3 \[FAQ Strange Chars\]](#), page 65)

### 8.2 Spurious Encodings

In rare cases, some commands (mostly from package `vc`) encode characters to tokens or even turn off X-Symbol mode. Package X-Symbol will not provide a workaround for these problems, because the situations in which they appear are too rare, the workarounds are easy, and the problems are not really caused by package X-Symbol.

- Doing the next logical version control operation (`C-x v v` and friends) encode characters to tokens when using package `crypt`.

Solution: use package `jka-compr` instead `crypt` (this is recommended anyway, see [Section 2.6.3 \[File IO Packages\]](#), page 9). Or kill the buffer and revisit the file.

- When using AucTeX with its default-mode algorithm, getting rid of the recently checked-in version of a file without reverting the buffer afterwards (`C-u C-x v c`) turns off X-Symbol mode without encoding the characters, e.g. under XEmacs/no-Mule, you see some strange characters like `'\233a'`.

Explanation: when using AucTeX's `TeX-default-mode`, the final `major-mode` is different from the initial `major-mode` deduced using `auto-mode-alist`. If this is the case, the VC command executes `normal-mode` which kills all local-variables including turning-off `x-symbol-mode`.

Solution: Turn on X-Symbol mode or change `auto-mode-alist` to directly choose `latex-mode`:

```
(push '("\\[tT]e[xX]\\'" . latex-mode) auto-mode-alist)
```

- When using AucTeX with its default-mode algorithm, writing a LaTeX buffer into a file with another file name turns off X-Symbol mode.

Explanation: Emacs sets the major mode with the file name. When using AucTeX's `TeX-default-mode`, we get the problems as described in the previous item.

Solution: Set `change-major-mode-with-file-name` to `nil` or use the solution from the previous item.

## 8.3 The Encoding Does Not Work

In a rare case, X-Symbol cannot do its encoding, i.e., convert the characters to tokens.

- `M-x write-region` fails to do the encoding if you use package `crypt`.

Explanation: with package `crypt`, the encoding has to be done by a function in `write-file-hooks` which is not used by `write-region`.

Solution: use package `jka-compr` instead `crypt` (this is recommended anyway, see [Section 2.6.3 \[File IO Packages\]](#), page 9). Or visit the region file and save it again via `C-x C-s`.

## 8.4 Frequently Asked Questions

It is assumed that you had successfully installed package X-Symbol, see [Section 2.11 \[Checking Installation\]](#), page 13.

### 8.4.1 XEmacs Crashes when using Input Method Token

It has been reported that XEmacs-21.0 to XEmacs-21.1.8 might produce cores when you use input method Token. That's why I strongly recommend to use XEmacs-21.1.9 or higher with package X-Symbol, see [Section 2.1 \[Requirements\]](#), page 5.

You get a warning during X-Symbol's initialization when using these XEmacs versions. If you don't want to upgrade, but also don't want to see the warning, you might want to set variable `x-symbol-xmas-warn-about-core` to `nil`.

A core in XEmacs always indicates a bug in XEmacs itself, not in a Lisp package like X-Symbol. Thus, send a bug report to the XEmacs team if you get cores with the *newest* version of XEmacs (please put me in the CC).

### 8.4.2 X-Symbol's Fontification does Not Work

In this case, super- and subscripts are not properly displayed (see [Section 8.4.4 \[FAQ No Subscripts\]](#), [page 65](#)) and under XEmacs/no-Mule, the buffer contains s.th. like ‘\233a’ (see [Section 8.4.3 \[FAQ Strange Chars\]](#), [page 65](#)). Possible causes:

- You have turned off `font-lock` or `font-lock` is out of sync. Use `M-x x-symbol-fontify`. See [Section 3.5 \[Role of font-lock\]](#), [page 22](#).
- The font-lock keywords of the current buffer are not prepared to display X-Symbol characters. See [Section 3.5 \[Role of font-lock\]](#), [page 22](#).
- You use package `fast-lock`. Solution: set `fast-lock-save-faces` to `nil` (done by default installation).
- You use some version control commands. You have probably noticed that these versions control commands also turn off `font-lock` in modes where you don't use X-Symbol, i.e., this is not a problem of package X-Symbol. See [Section 9.2.3 \[Wishlist Emacs\]](#), [page 78](#). See [Section 8.2 \[Spurious Encodings\]](#), [page 63](#).

### 8.4.3 The Buffer Contains Strange Characters

If you see s.th. like ‘\233a’, you see the internal representation of X-Symbol characters under XEmacs/no-Mule (see [Section 7.1 \[Char Representation\]](#), [page 51](#)) directly. Possible causes:

- You have `font-lock` problems, see [Section 8.4.2 \[FAQ font-lock\]](#), [page 65](#).
- More complicated editing commands like `C-t` may produce strange character sequences which do not represent X-Symbol characters, see [Section 8.1 \[Nomule Problems\]](#), [page 63](#).
- In some cases, e.g., when using the minibuffer for input via `M-%` or `C-s`, it would be too much work to fontify these character sequences in order to display proper X-Symbol characters. See [Section 8.1 \[Nomule Problems\]](#), [page 63](#).

If Emacs shows some strange glyphs for some characters in your buffers but not the Grid, there is a font in your font path which pretends to have charset registry-encoding `adobe-fontspecific`, but in fact uses another encoding. E.g., Mathematica's fonts cause the characters intersection and union to mix up. Possible solutions:

- Delete that font from the font path. Maybe moving it at the end also works.
- In Emacs-21, you have the chance to disable the use of some fonts (if you know something similar for XEmacs, please let me know). For example, to disable the fonts from Mathematica, use
 

```
(setq face-ignored-fonts '("\\"'-wri-math1"))
```
- If the characters show up correctly initially, but mix up after some font changing command, don't use that command. E.g., the font selection in XEmacs via the Options menu seems to lose some information about the original font. OK, this is not really a satisfying solution, but the whole issue isn't my fault, either.

### 8.4.4 I Cannot See any/some Super- or Subscripts

If you cannot select ‘Super-/Subscripts’ in the menu, the first of the following points is more likely the cause, the others otherwise.

- You have `font-lock` problems, see [Section 8.4.2 \[FAQ font-lock\]](#), [page 65](#).
- There are cases where super- and subscripts are not displayed, see [Section 5.1 \[Super and Subscripts\]](#), [page 33](#).



- The argument in braces are not correctly recognized, since the `font-lock` syntax-table is not correct. It should include ‘{’ as the only open parenthesis and ‘}’ as the only close parenthesis character. Note that this is quite difficult to archive under Emacs and XEmacs/Mule. This is a minor bug in the corresponding `font-lock` package, but would require other changes there, therefore not likely to be fixed. Fortunately, this does not happen often.

#### 8.4.5 I See Super- and Subscripts where I Don’t Want Them.

E.g., I see a subscript in arguments of `\label`. Package X-Symbol only uses super- and subscripts if they are in braces, if the `asciicircum/underscore` has not been fontified yet or is only fontified with faces which are allowed by `x-symbol-tex-font-lock-allowed-faces`, see [Section 5.1 \[Super and Subscripts\], page 33](#).

- You use the default `tex-font-lock-keywords`: The argument of `\include` and friends are not fontified by these, i.e., the use of super- and subscripts are not prohibited. Solution: add your own keyword for these commands or use package `font-latex`, see below.
- You use package `font-latex`. Solution: set `font-lock-maximum-decoration` to value `t`, `2` or higher. Package X-Symbol will still use subscripts in `\verb`, in the `verbatim` environment, in the argument of `\includegraphics` and probably other commands. Some of these problems will probably be solved by future versions of `font-latex`.
- You use my font-lock keywords (file ‘`x-font-lock.el`’): everything should work fine. Please note that this file is not meant to be a replacement of ‘`font-latex.el`’ useful to all users. Also, highlighting is a matter of taste, i.e., I am not going to change the ‘`x-font-lock.el`’ to support LaTeX-2.09, TeX’s math regions, other likings, etc.
- You use your own font-lock keywords for TeX. In this case, you be able to adapt the solutions from the previous points to your situation.

#### 8.4.6 The Characters are Too Small or Too Big

Why aren’t there more different font sizes? Because nobody (including the author) was in the mood to design them (actually only the `xsymb1` font needs to be designed). *Please do only ask the author whether they are in work if you are serious to do it yourself otherwise!*

Why do I get a lower-case letter when I should get a capital letter (or vice versa)? Please convince yourself (see [Section 5.3 \[Info\], page 37](#)) that you actually get the correct letter—they are just of different sizes. See [Section 2.9 \[Installing Fonts Lisp\], page 12](#).

I was told that the `xsymb1` font scales reasonably well to a larger font size—if you don’t think so, design a new font and send me the result.

#### 8.4.7 The Conversion Changes Some Tokens

In most token languages, a character might be represented by different tokens. If this character is encoded (when saving the buffer), the canonical representation is saved. See [Section 3.2.4 \[Unique Decoding\], page 18](#).

- Solution: Do not redefine standard TeX macros or use unique decoding.

#### 8.4.8 A Space is Added During the Encoding

A space is added after some characters during the encoding to tokens. With token languages `tex` and `utex` (not with language `sgml`), there must be a space after the token to recognize its end in some cases.



E.g., if your buffer contains ‘`a+b`’ (where `+` stands for the character `circleplus`), this is encoded to ‘`a\oplus b`’ (note the space after `\oplus`). Decoding it yields ‘`a+ b`’.

I admit, this looks ugly. The space is only added if the symbol character is followed by a letter or by ‘`@`’. Thus, decoding ‘`a\oplus\beta`’ yields ‘`a+b`’ (without space!).

- Suggestion: Also use a space before `\oplus`. The alternative would be to delete the space which other people won’t like.

For an exact description, See [Section 6.2.4 \[TeX Macro Conversion\]](#), page 45 for an exact description.

### 8.4.9 I Don’t Want 8bit Characters in the File

By default, these are not encoded if the buffer-local variable `x-symbol-8bits` is non-`nil`.

By default, this variable is only set to non-`nil`, if something like

```
\usepackage[latin1]{inputenc}
```

is found at the beginning of the file. That line does not make sense if you do not have 8bit characters in the file, i.e., delete it. See [Section 3.2.2 \[File Coding\]](#), page 16. Note: commenting the line is not enough! (I do not run LaTeX to check for the line, I just do plain text search.)

### 8.4.10 I Cannot Distinguish Character hyphen from ‘-’

In most fonts, the Latin character `hyphen` cannot be distinguish from the Ascii character ‘-’. If you do not want to decode the corresponding token `\-` or `&shy;`, put the following into your ‘`~/.emacs`’:

```
(setq x-symbol-tex-user-table '((hyphen)))
(setq x-symbol-sgml-user-table '((hyphen)))
```

A better alternative would be to make `font-lock` display these character in a different color.

### 8.4.11 Problems with Spell-checking

As explained in [Section 2.6.4 \[Miscellaneous Packages\]](#), page 10, `ispell` assumes the buffer contents to be the same as the file contents and does not provide any hook to fix this. This might break `ispell-word` and `ispell-region`. Possible symptoms:

- A word which contains letters which the program `ispell` does not know about is either not spell-checked or parts of it are spell-checked as independent words.

Solution: Use the `ispells` 8bit dictionaries even if you do not store 8bit characters in the file. This should fix the problem for almost every word, except, e.g., words containing the Latin-9 character `oe` if you use a Latin-1 encoding.

- Spell-checking might stop with the error message ‘`Ispell misalignment`’. I can reproduce this only with Emacs, not with XEmacs.

Question: If you know some settings (like for `process-coding-system-alist`) which solves this problem, please let me know!

Solution: turn X-Symbol off before spell-checking your buffer. This is of course no option if you use `flyspell`.

The real solution would be to fix `ispell`, at least by providing a useful hook which allows X-Symbol to fix the problem. See [Section 9.2.3 \[Wishlist Emacs\]](#), page 78. You are strongly encouraged to send a patch to the maintainer of `ispell`, you even get a paragraph here in [Section 9.4 \[Acknowledgments\]](#), page 80!

### 8.4.12 How to Use X-Symbol with Gnus or VM

You can also use X-Symbol to read and write your News and Mails. This sections includes coding for your ‘~/*.emacs*’ if you want to do so. It has been tested for Gnus-5.8.8 and VM-6.96; if you use RMAIL or MH-E, you have to try to find a solution yourself (please send it to me). Support for Gnus might become a standard part of X-Symbol.

```
(custom-set-variables
 '(x-symbol-auto-style-alist
   '(((mail-mode message-mode gnus-article-mode vm-presentation-mode)
      tex nil nil nil nil t nil))))
```

This is optional (you might want to use the Custom interface for the same effect) and tells Emacs/X-Symbol to use token language *tex* and to display super-/subscripts (if *font-lock* is enabled), X-Symbol is not automatically turned on. See [Section 3.3 \[Minor Mode\]](#), page 20.

```
(defun x-symbol-x-mail-send-hook ()
  (if x-symbol-mode (x-symbol-mode 0)))
(add-hook 'mail-send-hook 'x-symbol-x-mail-send-hook)
(add-hook 'message-send-hook 'x-symbol-x-mail-send-hook)
(add-hook 'vm-mail-send-hook 'x-symbol-x-mail-send-hook)
```

This tells Emacs to automatically turn off X-Symbol (which includes encoding characters to token) before actually sending the message.

```
(defun x-symbol-x-gnus-prepare ()
  (when x-symbol-mode
    (setq x-symbol-mode nil)
    (x-symbol-mode-internal nil)))
(add-hook 'gnus-article-prepare-hook 'x-symbol-x-gnus-prepare)
```

Since Gnus reuses the ‘*\*Article\**’ buffer, where X-Symbol could have been turned on previously, we must make sure that X-Symbol is turned off with the new article.

```
(defun x-symbol-x-vm-prepare ()
  (and (boundp 'vm-presentation-buffer)
       (buffer-live-p vm-presentation-buffer)
       (save-excursion
        (set-buffer vm-presentation-buffer)
        (when x-symbol-mode
          (setq x-symbol-mode nil)
          (x-symbol-mode-internal nil)))))
(add-hook 'vm-select-message-hook 'x-symbol-x-gnus-prepare)
```

The same thing for VM, although the hook is not as nice as Gnus’ one; the function therefore might depend a bit too much on VM’s interna.

```
(put 'vm-mode 'x-symbol-mode-disable
     "Use VM Presentation Mode to turn on X-Symbol")
(custom-set-variables '(vm-fill-paragraphs-containing-long-lines 80))
```

You cannot use X-Symbol in VM Mode, only in VM Presentation Mode (X-Symbol would change your ‘*INBOX*’). The first (optional) Emacs Lisp expression gives you a better error message when you try to turn on X-Symbol Mode in VM Mode. The second line makes sure that VM always uses VM Presentation Mode to display the articles.

## 8.5 How to Send a Bug/Problem Report

Bug fixes, bug/problem reports, improvements, and suggestions are strongly appreciated. So are corrections to this manual (better explanations, correcting my English, ...). Especially

useful would be some feedback by people using default fonts with a charset registry-encoding other than `iso8859-1` (Western encoding).

Please read this section carefully, even if you generally know how to send a bug report (see [section “Bugs” in XEmacs User’s Manual](#)). This might look tedious to you, but it actually saves a lot of time (your time, too).

The **general recommendation** for bug/problem reports is: give the impression that you really have tried to find the necessary information yourself and make your report precise while including all information you have.

For each bug/problem report or question you want to send to the maintainer, please use the following sequence:

1. Make sure that you use the **newest version** of X-Symbol. You are reading Edition 4.5.1 (XEmacs) of the manual for X-Symbol 4.5.1.
2. Read the manual, especially [Section 2.11 \[Checking Installation\]](#), [page 13](#), [Chapter 8 \[Problems\]](#), [page 63](#), and [Section 8.4 \[FAQ\]](#), [page 64](#). The four indexes (see [\[Indexes\]](#), [page 83](#)) might also lead you to an answer to your question.

3. Use `M-x x-symbol-package-bug` (also to be found in X-Symbol’s Command submenu) to write your report describing *one* bug or problem, i.e., use **different mails** for **unrelated problems**. Please do not “reuse” a mail thread with the maintainer, i.e., if you start a section with “Here is another problem”, you do something wrong.

If Emacs is not your mail tool, copy the Subject header line and the message body from Emacs’ `*mail*` buffer to your mail tool.

If `M-x x-symbol-package-bug` fails to work, you have a problem with your installation and your report should be about this problem. In this case, use `‘x-symbol version; summary’` as Subject header where *version* is the version of X-Symbol (it should be 4.5.1) and *summary* is a brief summary of your installation problem.

(*Rationale:* This command automatically extracts some essential information without any work by you. Don’t waste your time pondering whether you should really use this command to write your report.)

4. Start your report with:

In the manual, I checked the sections *section1*, *section2*, ..., but didn’t find anything which helped me with the following problem:

The sections *section1*, *section2*, etc are names of the sections (not whole chapters) in the manual where you would expect an answer to your question/problem/bug.

If you didn’t know which sections to inspect, please check the indexes. If they are not helpful, send me words/terms which should be included in the indexes.

(*Rationale:* This way, I get an idea where to improve the manual, especially by adding cross references.)

5. If buffer `‘*Warnings*’` does not exist in the buffer menu, everything is fine so far. So is (for me as the author of package X-Symbol), if `‘X-Symbol’` is not mentioned there. Otherwise, include the contents of buffer `‘*Warnings*’` into your bug report.

Temporary Emacs (< v21.4) note: the warnings might be somewhere hidden in buffer `‘*Messages*’`; please check that buffer.

6. Put the parts of the code from `‘~/ .emacs’` and the system-wide files which causes the problem into a fresh file `‘my-problem.el’`. The problem/error should be visible when invoking

`xemacs -no-site-file -q -l my-problem.el`

In the minimal case, `‘my-problem.el’` just contains the following line (see [Section 2.4 \[Installing Lisp\]](#), [page 6](#)):

`(x-symbol-initialize)`

If the error has disappeared after you have included your complete `'~/.xemacs/init.el'` and `'~/.emacs'`, the problem is likely caused by some code of your system-wide installation. Include the code, which can be found using command `M-x find-library` with files `'site-start'` and `'default'` (everything is fine if these files do not exist).

If you use `'x-symbol-site.el'` (its use is deprecated), copy its contents into `'my-problem.el'` and delete the corresponding load command.

Attach the file `'my-problem.el'` to your report. **Please try to minimize the size of `'my-problem.el'`!** A standard technique is recursive halving: Delete the second half of `'my-problem.el'`. If the problem disappears, delete the first half instead. Do the same with the smaller file again, ....

(*Rationale:* Most problems are a consequence of some specific customizations, but I don't have time to debug each user's init file.)

7. If you have set variable `custom-file` in `'my-problem.el'`, attach the corresponding file to your report.
8. If the error can only be reproduced in combination with another Emacs package, please send me:
  - If it is included in the standard Emacs/XEmacs distribution / if is an XEmacs package: the version you use if it is not that from the Emacs/XEmacs distribution (use `M-x find-library` to check whether you really use the version from the Emacs/XEmacs distribution).
  - If it is a non-standard (and non-obscure) package: the URL of the distribution and/or the source.
  - Otherwise: include its code into `'my-problem.el'` and delete the corresponding load or `require` command. Then, reduce the size of `'my-problem.el'` as described above.
9. If the problem is not reproducible with an *arbitrary* (`'.tex'`, `'.html'`, ...) file, include the file with its full file name into your bug report. (If you like, you can try to minimize the file if the problem is still reproducible.)

(*Rationale:* Most problems are only reproducible with specific files.)

10. Finally, include the exact key sequence which causes the problem into your bug report. You should also tell me the name of the buffer in which the problem occurred and how you have created that buffer (e.g., by `C-x C-f file` `(RET)`).

At best, you start your Emacs, and then try to reproduce the problem as fast as possible (i.e., with a minimum number of key/mouse strokes).

As soon as the problem appears, press `C-h l` and include the contents of buffer `'*Help*'` in your bug report.

(*Rationale:* Most problems are only reproducible with point being at a specific position in the file, with specific key sequences, etc.)

11. If you have problem with the display of images, please include the output of the shell commands `'convert -h'` and `'convert -list Format'` in your bug report. If the first command fails, you have a problem with the program `convert`, not X-Symbol.
12. If necessary, include a screen-shot in your bug report.
13. If you could not use `M-x x-symbol-package-bug`, include the contents of buffer `'*Help*'` after the following actions:
  - Type `C-h v x-symbol-version` `(RET)`.
  - Type `C-h v emacs-version` `(RET)`.
  - Type `C-h v features` `(RET)`.

If you have solved your problem during this sequence, but you think your situation is worth to be mention in this manual (e.g., in [Section 2.6 \[Package Integration\]](#), [page 7](#)), I would appreciate if you would send me a some new text for this manual or a normal bug report together with your solution.



## 9 History and Projects

### 9.1 News: Changes in Recent Versions of X-Symbol

This is the complete history of X-Symbol. It just lists the major changes before Version 3.0.

#### 9.1.1 Changes in X-Symbol 4.5.1

Version 4.5.1 has not yet been announced.

- Various bug fixes and minor changes.

#### 9.1.2 Changes in X-Symbol 4.5

Version 4.5 has been released on March 2003 as beta.

- X-Symbol finally respects the Mule coding system of each individual buffer.
- Bug fix: would mess up encoding of math-mode characters with token language `bib`. Other conversion fixes for languages `bib` and `texi`.
- Bug fix (workaround for bug in XEmacs): auto-save files would have length 0.
- Bug fix (Emacs): package now works with package `crypt/crypt++`.
- Token language `sgml`: always encode characters to entity references by default (where defined by the HTML standard). Include `hm--html-mode`, `html-helper-mode`, remove `sgml-mode` as typical major modes which use X-Symbol.
- Token language `tex`: support some symbols of package ‘`stmaryrd.sty`’.
- Change the auto-style, formerly auto-mode, mechanism.
- Image support when running on Emacs.
- New input method Quail, a usual Mule input method.
- Corrected Latin-5 definitions. Support Latin-5 (“Turkish”) on XEmacs running under Windows.
- X-Symbol works with Emacs/XEmacs running under a character terminal.
- Improvements for external languages. Super-/subscript matching of token languages has changed.
- X-Symbol can use package `format` and does not require special fonts for super-/subscripts with Emacs-21.4+. Still open whether this will be used. . . .
- Dropped support for XEmacs-20.3.
- Various bug fixes and minor changes.

#### 9.1.3 Changes in X-Symbol 4.2 to 4.4

Version 4.4 has been released on June 2002 as beta.

- Token language TeX has changed: no excessive use of braces anymore, no excessive normalization, and aware of environments `{tabbing}` and `{verbatim}`, and macro `\verb`. Reading and saving “old-encoded” files works without changes in the file (the buffer looks different), there is also a command to remove the unwanted braces around accented letters.
- New token language “BibTeX Macro” (`bib`, similar to old `tex`), used for BibTeX files.
- Nuked executables, the Lisp conversion for all languages is now 2-5 times faster.



- Latin-9 support. Latin-9 font included in distribution.
- Works with XEmacs-21.4+ on Windows. Of course, it just supports a limited number of characters and no super- and subscripts there due to missing fonts.
- More likely to save 8bit characters in the file by default: also look for 8bit characters in the file when visiting the file, also inspect master file (`TeX-master`) with token language `tex`.
- New buffer-local variable `x-symbol-unique`: when non-`nil`, decodes much less tokens to avoid near to all normalizations, used for `TeX`'s style files (but X-Symbol is not automatically turned on). Dropped token language `utex`.
- Menu changes, new commands: submenu “Conversion”, menu items “Copy Encoded”, “Paste Decoded” and others.
- Special coding for `preview-latex`. Using X-Symbol now only gives a 10% overhead of `previews` parsing time.
- X-Symbol now works with Whizzy`TeX`.
- The interface for defining a token language has changed, it is also much more general, useful for ProofGeneral.
- Changed final bytes of ISO 2022 escape sequence for X-Symbol charsets since Emacs reserves the characters ‘0–9’ for itself. Does XEmacs has any policy here (it also uses ‘?’)?
- Dropped workaround for minor bug in XEmacs-20.X.
- Various bug fixes and minor changes.

#### 9.1.4 Changes in X-Symbol 4.1

Version 4.1 has been released on Mar 2002 as beta.

- X-Symbol works with Emacs-21.1 or higher. Porting is not complete, yet.
- New token language “`TeXinfo` command” (`texi`).
- Slightly different definition of “valid character”.
- Remove the “local if set” and “default: ...” submenu stuff.

#### 9.1.5 Changes in X-Symbol 3.4

Version 3.4 has been released on Mar 2002.

- Moved to SourceForge.net. Added files for nicer HTML output of manual.
- Would sometimes perform strange conversions when `global-flyspell-mode` is enabled.
- Bug fixes: command `M-x write-region` would always save the whole buffer if X-Symbol is enabled for that buffer, writing a remote file via `ange-ftp` would not work (was OK with `efs`).
- Automatically deduce default coding via `locale -ck LC_CTYPE`.
- Issue warning when running on XEmacs-21.0 to XEmacs-21.1.8. Update manual: XEmacs user package directory is `~/xemacs/packages`.
- Directories ending with `/**` in image search paths are recursive.
- New characters used for token languages “`TeX macro`” and “`Isabelle symbol`”.
- Make sure to convert just the first part of a multi-part image.
- Source distribution includes files for building an RPM package, all files also compile without Mule support.
- Minor changes. Manual changes.

### 9.1.6 Changes in X-Symbol 3.3

Version 3.3 has been released on Jan 1999.

- Package X-Symbol is really a proper XEmacs package: no need to create fonts and to set the font path. With XEmacs/no-Mule, I still recommend to create the executables (type `M-x x-symbol-exec-create`).
- New functions used for interaction with Emacs package `comint`. This is necessary for new token language “Isabelle symbol”, to be distributed with Emacs package `ProofGeneral`.
- New characters used for token languages “ $\TeX$  macro” and “Isabelle symbol”.
- Minor changes. Manual changes.

### 9.1.7 Changes in X-Symbol 3.2

Version 3.2 has been released on Dec 1998.

- Package X-Symbol is a proper XEmacs package. The installation process is much easier (using the binary package). It has changed, though! The use of file ‘`x-symbol-site.el`’ is deprecated.
- Reverting the buffer and using `vc` commands do not encode characters when not using `crypt`. (This did not work always.)
- Workaround for bug (segfault) in XEmacs-21/Mule betas.
- Command `x-symbol-package-bug` is less restrictive. Please use this command to contact the maintainer.
- Bug fixes. Minor changes. Manual changes.

### 9.1.8 Changes in X-Symbol 3.1

Version 3.1 has been released on Oct 1998.

- $\TeX$  macro `\mu` is represented by a character in the Adobe Symbol font, not in a Latin-`{1,3,5}` font anymore.
- Support for most SGML entities in HTML-4.0 specification.
- Additional characters for `\therefore`/`&there4;`, `&oline;` and `&euro;`.
- Package X-Symbol has been customized.
- The documentation has been completed (as  $\TeX$ info file).
- Handle special URL prefixes ‘`file:`’, ‘`http:`’ for images.
- Bug fixes, configuration changes.

### 9.1.9 Changes in X-Symbol 3.0

Version 3.0 has been released on Sep 1998 as beta.

- Package X-Symbol now works on XEmacs with and without Mule support. Dropped support for XEmacs 19.13 to 19.16/20.2.
- Full support of token language `sgml` (executables, subscripts, images).
- X-Symbol is a proper minor mode.
- Easier (automatic) 8bit character control (e.g., for `\times \pm, \dots`). By default, the encoding when saving only writes 8bit characters, if ‘`\usepackage[latinn]{inputenc}`’ with `n=1,2,3,5` was found in the first 10000 characters of the file (including commentary).

- Package X-Symbol can be easily extended with new token languages and fonts due to its modular design. It consistently handles situations where an entry for an additional character defines the same preferred key binding (and context) as for a previously defined character
- Key bindings have completely changed. They are now consistent with the contexts of input method Context (which have changed a bit).
- The keys @ and ! are not used anymore as Modify- and Rotate-Key. The Rotate key (instead of the Modify-Key) is used to “Greek”ify the previous Ascii char.
- Input method Aggressive Context is now called input method Electric and is much more restrictive (using package `texmathp` with language “`TEX macro`”).
- Easier installation despite many additional features.
- Supports more characters.
- Nicer grid, info in echo area.
- Better cooperation with packages: `vc` (check-out does not convert characters), `reftex` (no strange characters ‘\237’, help with label creation), `auctex`, `ispell`, `font-latex` (no annoyances with `\exists`).
- Safer use of executables.
- The code has completely changed. You have to redo your installation.

### 9.1.10 Changes in Old Releases.

This sections gives just an overview of the major changes in the releases.

Version 2.6 has been released on Oct 1998.

- Fixed serious bug when used under tty.

Version 2.5 has been released on Mar 1998.

- Image support.

Version 2.4 has been released on Mar 1997.

- Token language `sgml`. (X-Symbol can handle more then token language `tex`.)
- Input method Aggressive Context (precursor of input method Electric), input method Context has been much improved.
- Fixed performance bug when saving a file with package `crypt`.
- Control of Conversion and 8bit character has changed.

Version 2.3 has been released on Sep 1996.

- Distributed with own font for more math characters.
- Info for the character around point in echo area.

Version 2.2 has been released on June 1996.

- Input method Grid. Help when using input method Keyboard.
- Control of Conversion and 8bit character has changed.

Version 2.1 has been released on April 1996.

- Fixed serious performance bug when loading files with `font-lock/lazy-lock`. Use executables for conversion of large buffers.
- The package `iso-cvt` is not integrated anymore. Now this package can also convert to/from Latin-1 characters, it is much faster.
- Menu support, including input method Menu.
- `isearch` works with X-Symbol characters.

- First multi-file version.

Version 1.4 has been released on Feb 1996.

- Provide some kind of “poor man’s Mule” to remove most Nomule-Problems.

Version 1.3 has been released on Jan 1996.

- Input method Abbrev (precursor of input method Token).
- Super-/subscript support.

Version 1.2 has been released on Jan 1996. It was the first release.

- Conversion between characters and  $\text{\TeX}$  tokens. Do so automatically when visiting a file and saving the buffer.
- Input method Keyboard.

## 9.2 Wishlist: Projects for X-Symbol

You are encouraged to try to provide a solution to one of the problems of this section. In fact, it is quite unlikely that I do it myself without any contributions from you, see also [Section 9.3 \[Open Questions\]](#), page 79.

Providing a solution to these problems is the second way of making your name appear in [Section 9.4 \[Acknowledgments\]](#), page 80.

### 9.2.1 Wishlist: Additional Token Languages

Making a contribution here would require just a basic knowledge of Emacs and X-Symbol. In fact, I would do the non-trivial part of the Emacs Lisp part (see [Section 7.4 \[Extending X-Symbol\]](#), page 55) for general-interest token languages (e.g.,  $\text{\AmsTeX}$ ).

It is likely that this would require additional fonts: available fonts (e.g., IPA font), handcrafted, or generated (see [Section 9.2.2 \[Wishlist Fonts\]](#), page 77).

### 9.2.2 Wishlist: Generated Fonts

A specific direction of font generation would be from ‘.bdf’ or ‘.pcf’ font files to Windows fonts to get rid of the limited support for XEmacs on Windows (see [Section 2.1 \[Requirements\]](#), page 5). If you have successfully converted X-Symbol’s fonts from the Unix format to the Windows format (via `bdftofon` or whatever) or if you have free and real Latin-N fonts for Windows, please *let me know*! I would also appreciate if you would actively try to get those missing Windows fonts.

The general direction is to automatically generate the ‘.bdf’ or ‘.fon’ fonts from other sources. This would have various advantages:

- We could easily create different sizes for our symbol font.
- It would be quite simple to create a font for  $\text{\AmsTeX}$  macros, etc., which would be displayed as X-Symbol characters by package X-Symbol.
- We could easily create different sizes for our symbol font.
- We would have fonts for both X11 and Windows.

New fonts for X-Symbol are being worked on. You can find material to generate them at the [web pages of X-Symbol](#). Quite a few problems needs to be fixed though, so it is considered as experimental. You are welcome to try, fix and report on the [X-Symbol development mailing list](#).

General open design issues (i.e., they could be re-thought for the currently used handcrafted fonts, too) are:

- Different  $\text{\TeX}$  macros (same appearance, different  $\text{\TeX}$  class = different spacing) use the same MetaFont character, e.g., `\dagger` and `\dag`. Therefore, we need different X11 characters for them.
- Some Ascii characters have a special meaning in  $\text{\TeX}$ . The corresponding MetaFont character is therefore produced by a  $\text{\TeX}$  macro, e.g., ‘{’ by `\{`. We need a X11 character which looks similar to the character but not exactly like it.

We could ask the question whether we should really distinguish the characters by appearance...we have the minibuffer info for the X-Symbol character anyway.... Here are the options:

- distinguished by size/underlining/miscellaneous (currently used),
- distinguished by different spacing (my current favorite),
- not distinguished

### 9.2.3 Wishlist: Changes in Emacs/XEmacs

Changes in Emacs and/or XEmacs would improve package X-Symbol, too:

- In Emacs: a package system similar to XEmacs’ one. The installation would be easier.
- The package `ispell` assumes the buffer contents to be the same as the file contents and does not provide any hook to fix this. This should be fixed in `ispell` (it will be better in Emacs-21.4), see [Section 2.6.4 \[Miscellaneous Packages\]](#), page 10.
- Some versions control commands turn off `font-lock`. This should be changed.
- Provide a face property `raise`: we wouldn’t need extra fonts for super- and subscripts. Emacs: it’s already a display property, make it a face property, too (or make `font-lock` set properties other than faces). XEmacs: no such property, yet.
- You are sometimes unnecessarily asked (because X-Symbol will encode the corresponding characters anyway) for a safe coding system. In Emacs (will be fixed in Emacs-21.4) for non-default Latin characters. In XEmacs, for all non-default characters if you use package `latin-unity` (see [Section 2.6.3 \[File IO Packages\]](#), page 9).
- In Emacs, will be fixed in 21.4. Using `isearch` and the input method Grid would not work.
- In XEmacs, fixed in 21.X. In `after-insert-file-functions`, there should be a possibility to get to know the start position of the region which is inserted. If `insert-file-contents` is called with argument `replace` being non-nil, it is not always point.
- In Emacs and XEmacs, will be fixed in Emacs-21.4. Make possibility to change buffers in `write-region-annotate-functions` official, see [Section 7.6.3 \[Alt Auto Conversion\]](#), page 61, have a way to get the original buffer.
- Since `font-lock` uses duplicable text properties in some cases, I need a function like `insert-buffer-substring-without-extents`. (Currently, I remove the extents afterwards, which looks slow for me.)
- In XEmacs. Run hooks in `post-command-hook` even if command exits with an error or quit (as it is in Emacs) or having some `post-error-or-quit-hook`. See [Section 8.1 \[Nomule Problems\]](#), page 63.
- In XEmacs. There are some bugs in package `custom/widget` (still in XEmacs-21.4) which are visible during the customization of X-Symbol.

### 9.2.4 Wishlist: Changes in LaTeX

Changes in LaTeX, especially ‘`inputenc.sty`’, would improve package X-Symbol, too:

- To make the definition of the character U00B5 consistent with Unicode, ‘`inputenc.sty`’ should define the character to stand not for the token `\mu` (U03BC is the right character), but for an extra token, e.g., something like `\textmicro`. X-Symbol uses `\mathmicro` here in order to avoid changing `\mu` to the character U00B5 if you have chosen to store 8bit characters.
- Use same encoding for both text and math, i.e. use `periodcentered` for both `\textperiodcentered` (the default) and `\cdot`. At least provide text-and-math versions for characters where no alternative is more obvious than the other. If that is not possible, always choose text mode except for `\not`, `\pm`, `\times` and `\division`: use `\textonesuperior` for U00B9, `\texttwosuperior` for U00B2, and `\textthreesuperior` for U00B3.
- The T<sub>E</sub>X macros `\textcent`, `\textcurrency`, `\textbrokenbar`, `\textyen` are defined as not available with OT1 and T1 font encoding. This should be changed.

### 9.2.5 Various Projects for X-Symbol

The following suggestions seem to be useful, though not essential:

- It would be nice if we could print the buffer contents. Currently, you see strange characters instead X-Symbol’s own characters.

Printing non-standard fonts is only possible via the Emacs package `ps-print`. A newer version of `ps-print` might be probably already capable of doing it. Thus, you are encouraged to help the XEmacs team updating this package.

### 9.2.6 Rejected Suggestions for X-Symbol

The following suggestions seem to be not useful enough to be worth the additional effort and increased package size. I might be convinced otherwise by patches (i.e., code, not text), though:

- It would be nice if X-Symbol would replace the token with the last character of the token if this is possible (see [Section 4.2 \[Input Method Token\]](#), page 26), not just with the next character. Well, during typing, this is not really annoying and after a while, you will use input method Token only for very short tokens.

## 9.3 Open Questions

This section lists some minor open questions.

- Loading file ‘`x-symbol.el`’ will initialize package X-Symbol (via function `x-symbol-initialize`), since all functions will need the initialization. In my opinion, this is no problem, since all customization options are defined in other files which do not require file ‘`x-symbol.el`’. Thus, customizing package X-Symbol will not initialize package X-Symbol.

The alternative would be to call function `x-symbol-initialize` in every function which can be autoloaded. This seems quite tedious to me. Also, I do not see a reason not to call `x-symbol-initialize` top-level in file ‘`x-symbol.el`’. If I am wrong here, please let me know (with an explanation). Batch-compilation might be an issue...

- When is necessary to set `x-symbol-auto-conversion-method` to `slowest`? Of course, it is only necessary when using `crypt`. Is the other necessary condition to use the computer pool of the University of Edinburgh?



## 9.4 Acknowledgments

Stefan Monnier did many of the changes necessary for porting X-Symbol to Emacs-21. Fortunately, he not only changed X-Symbol to use a quite different API on Emacs for things like charsets and menus, he also made the necessary changes in Emacs itself. Before that, Sang-Min Lee started porting X-Symbol to Emacs-20.4, which was important for moving the status of the Emacs port of X-Symbol from “todo” to “in work”.

David Kastrup demonstrated that the old way of encoding characters to  $\TeX$  macros generally inhibited ligatures and kerns, i.e., it was worse than expected. He also discussed the details of how to do the encoding and decoding right. Christophe Raffalli suggested to use a decode method which can be used for a larger class of token languages. He also proved that it is faster.

Solofo Ramangalahy is working on scripts to generate X-Symbol fonts from other sources. This has various advantages and is discussed in more detail at [Section 9.2.2 \[Wishlist Fonts\]](#), page 77. His work is now available at the X-Symbol download area.

Package `math-mode` by Renaud Marlet and the extension of it by Julian Bradfield gave the basic idea for the following features: supporting  $\TeX$ ’s math macros, input methods token, context/electric, super-/subscript support. The shell script `makesub` is a merge and change of the scripts `makesupers` and `makesub` by Julian.

The font ‘`xsymb0`’, which is distributed with this package, is a minor modification (appearance) of the Adobe symbol font, thanks to its non-restrictive copyright. You may use the Adobe font instead. The special images are from package `frame-icon`.

The idea for Help during an X-Symbol key sequence is from package `x-compose`. The general idea for showing some info in the echo area is from package `eldoc`. The trick which stops `expand-abbrev` is from package `mail-abbrevs`. The idea for `x-symbol-image-cache-directories` is from package `fast-lock`. The code for image command parsing is influenced by some code in package `font-lock`. The code around `x-symbol-image-delete-extents` is based on some code in package `bib-cite`.

*Thanks for patches/reports/suggestions to:* Vladimir Alexiev, David Aspinall, Masayuki Ataka, Neal Becker, Matthias Berberich, Stefano Bianchi, Janusz S. Bien, Uwe Brauer, Alastair Burt, John Collins, Laurent Descamps, Frederic Devernay, Carsten Dominik, Steve Dunham, Michael Ebner, Stephen Eglen, Paul Furnanz, Jeffrey Grandy, Clemens Gröpl, Kenichi Handa, Meik Hellmund, Rjurick M. Hristev, Adriaan Joubert, Marcin Kasperski, David Kastrup, Richard Ketchersid, Felix E. Klee, Gerwin Klein, Thomas Kleymann, Ekkehard Koehler, Fred Labrosse, Jan-Ake Larsson, Bernhard Lehner, Stefan Monnier, Harald Muehlboeck, Karsten Muehlmann, Jakub Narebski, Peter Møller Neergaard, Raymond Nijssen, David von Oheimb, Alex Ott, Sudeep Kumar Palat, Arshak Petrosyan, Jim Radford, Christophe Raffalli, Solofo Ramangalahy, Alex Russell, Marciano Siniscalchi, Richard M. Stallman, Axel Thimm, Eli Tziperman, Jan Vroonhof, Markus Wenzel, Sabine Wetzl, Pierre-Henri Wullemmin, Roland Zumkeller, Marco Zunino, Gerard Zwaan.

*Thanks for general information to:* Per Abrahamsen, Steve L. Baur, Kenichi Handa, David Kastrup, Gerd Moellmann, Stefan Monnier, Primoz Peterlin, Martin Ramsch, Peter Schmitt, Toby Speight, Jan Vroonhof, Eli Zaretskii.

I made use of information from the following URLs:

```
http://www.w3.org/TR/REC-html40/sgml/entities.html
http://www.fmi.uni-passau.de/~ramsch/iso8859-1.html
http://czyborra.com/charsets/iso8859.html
http://www.bbsinc.com/iso8859.html
http://www.bbsinc.com/iso8879.html
http://ppewww.ph.gla.ac.uk/~flavell/charset/internat.html
http://ppewww.ph.gla.ac.uk/~flavell/iso8859/iso8859-pointers.html
```



<http://sizif.mf.uni-lj.si/linux/cee/iso8859-2.html>

I do not intend to update this list in the future—this is just an "Acknowledgment" section.



## Indexes

You should consult the following indexes if you are interested in a specific feature or aspect of package X-Symbol. You should also consult them before sending a report to the maintainer (see [Section 8.5 \[Bug Reports\]](#), page 68),

The links lead you to the manual sections describing X-Symbol's commands and variables. See [Section 1.3 \[About\]](#), page 2.

## Key Index

### ?

? ..... 27

### B

[button1](#) ..... 28

[button2](#) ..... 27, 28, 37

[button3](#) ..... 27, 28, 37

### C

C-, ..... 29

C-. ..... 29

C= ..... 25

C= C-= ..... 27

C= C-h ..... 28

C= [down](#) ..... 29

C= [help](#) ..... 28

C= [left](#) ..... 29

C= [RET](#) ..... 26

C= [right](#) ..... 29

C= [TAB](#) ..... 26

C= [up](#) ..... 29

C-h ..... 28

### H

h ..... 27

[help](#) ..... 28

### I

i ..... 27

### M

M-[end](#) ..... 28

M-[home](#) ..... 28

M-[next](#) ..... 28

M-[prior](#) ..... 28

[multi-key](#) ..... 25

### Q

q ..... 27

### R

[RET](#) ..... 27

### S

[SPC](#) ..... 27

## Program and Package Index

### A

abbrev ..... 10  
 ‘amssymb.sty’ ..... 46  
 ange-ftp ..... 9  
 auctex ..... 5, 8, 64

### B

bdftofon ..... 77  
 bib-cite ..... 8  
 bibtex ..... 49

### C

comint ..... 9  
 completion ..... 10  
 convert ..... 5, 7, 33, 35  
 crypt ..... 9, 64, 79  
 crypt++ ..... 9

### D

desktop ..... 10  
 display ..... 37

### E

efs ..... 9  
 Emacs ..... 5  
 Exceed ..... 11

### F

fast-lock ..... 8  
 flyspell ..... 10  
 font-latex ..... 9, 66  
 font-lock ..... 5, 9, 51, 65, 66  
 ‘fontenc.sty’ ..... 46  
 format ..... 61  
 frame-icon ..... 80  
 func-menu ..... 10

### G

GNU texinfo ..... 50  
 Gnus ..... 68

### I

‘inputenc.sty’ ..... 46, 78  
 iso-cvt ..... 9  
 iso-sgml ..... 9  
 ispell ..... 10, 67, 78

### J

jka-compr ..... 9

### L

latex2html ..... 5, 13  
 ‘latexsym.sty’ ..... 46  
 latin-unity ..... 9, 20  
 lazy-lock ..... 9  
 lazy-shot ..... 5, 9

### M

makeinfo ..... 5, 13, 50  
 math-mode ..... 80  
 Mathematica ..... 65  
 Message ..... 68

### N

Netscape ..... 47

### P

perl ..... 11  
 preview-latex ..... 8  
 ProofGeneral ..... 10, 50  
 ps-print ..... 79  
 psgml ..... 10  
 psgml-html ..... 10

### R

reftex ..... 8, 38

### S

session ..... 10  
 ‘stmaryrd.sty’ ..... 46

### T

texi2dvi ..... 5, 13  
 texi2html ..... 50  
 texinfo ..... 50  
 texmathp ..... 5, 8, 42

### U

ucs-tables ..... 10

### V

vc ..... 10, 63, 78  
 VM ..... 68

### X

x-compose ..... 10  
 XEmacs ..... 5  
 xfd ..... 11  
 xfig ..... 37  
 xfontsel ..... 11  
 xset ..... 11

## Command, Function and Variable Index

### A

after-insert-file-functions ..... 61, 78

### B

backward-char ..... 59  
backward-char-command ..... 59

### C

change-major-mode-with-file-name ..... 64  
comint-input-sender ..... 9

### F

fast-lock-save-faces ..... 8, 65  
font-lock-auto-fontify ..... 9  
font-lock-maximum-decoration ..... 9  
forward-char ..... 59  
forward-char-command ..... 59

### G

global-flyspell-mode ..... 10  
gnus ..... 68  
gnus-article-prepare-hook ..... 68

### H

hack-local-variables-hook ..... 61

### I

isearch ..... 78  
ispell-region ..... 67  
ispell-word ..... 67

### K

kill-ring ..... 19

### L

LaTeX-math-insert-function ..... 8

### M

mail ..... 68  
mail-send-hook ..... 68  
message-mail ..... 68  
message-send-hook ..... 68

### N

newline ..... 59  
newline-and-indent ..... 59

### P

post-command-hook ..... 78

### R

reftex-translate-to-ascii-function ..... 8  
reindent-then-newline-and-indent ..... 59

### S

save-buffer ..... 9  
self-insert-command ..... 59  
sgml-close-angle ..... 59  
sgml-slash ..... 59

### T

TeX-insert-dollar ..... 59  
TeX-insert-punctuation ..... 59  
tex-insert-quote ..... 59  
TeX-insert-quote ..... 59  
TeX-master ..... 8, 41, 43  
TeX-next-error ..... 8  
TeX-region-hook ..... 8  
TeX-translate-location-hook ..... 8  
TEXINPUTS ..... 43  
TEXPICTS ..... 43

### U

unify-8859-on-decoding-mode ..... 10  
unify-8859-on-encoding-mode ..... 10

### V

vc-next-action ..... 10  
vm ..... 68  
vm-mail ..... 68  
vm-mail-send-hook ..... 68  
vm-mode ..... 68  
vm-presentation-mode ..... 68  
vm-select-message-hook ..... 68

### W

write-file ..... 64  
write-file-data-hooks ..... 61  
write-file-hooks ..... 61, 64  
write-region ..... 64  
write-region-annotate-functions ..... 61, 78

### X

x-symbol-8bits ..... 17  
x-symbol-after-init-input-hook ..... 31  
x-symbol-auto-8bit-search-limit ..... 18  
x-symbol-auto-coding-search-limit ..... 17  
x-symbol-auto-conversion-method ..... 9, 79  
x-symbol-auto-key-autoload ..... 25

x-symbol-auto-mode-suffixes .....	21	x-symbol-image-temp-name .....	36
x-symbol-auto-style-alist .....	21	x-symbol-image-update-cache .....	36
x-symbol-bib-auto-style .....	49	x-symbol-image-use-remote .....	36
x-symbol-bib-class-alist .....	49	x-symbol-init-language-interactive .....	61
x-symbol-bib-class-face-alist .....	49	x-symbol-initialize .....	6, 7, 79
x-symbol-bib-electric-ignore .....	49	x-symbol-installer-address .....	6, 39
x-symbol-bib-extra-menu-items .....	49	x-symbol-key-min-length .....	55
x-symbol-bib-header-groups-alist .....	49	x-symbol-key-suffix-string .....	54
x-symbol-bib-modes .....	49	x-symbol-lang-auto-style .....	21
x-symbol-bib-user-table .....	49	x-symbol-lang-class-alist .....	23
x-symbol-character-info .....	37	x-symbol-lang-class-face-alist .....	23
x-symbol-charsym-ascii-alist .....	38	x-symbol-lang-electric-ignore .....	30
x-symbol-charsym-ascii-groups .....	38	x-symbol-lang-extra-menu-items .....	22
x-symbol-coding .....	16	x-symbol-lang-header-groups-alist .....	23
x-symbol-compose-key .....	25	x-symbol-lang-image-cached-dirs .....	36
x-symbol-context-ignore .....	29	x-symbol-lang-image-keywords .....	34
x-symbol-context-info .....	38	x-symbol-lang-image-searchpath .....	34
x-symbol-context-info-ignore .....	38	x-symbol-lang-master-directory .....	34
x-symbol-context-info-ignore-groups .....	38	x-symbol-lang-modes .....	21
x-symbol-context-info-ignore-regexp .....	38	x-symbol-language .....	15
x-symbol-context-info-threshold .....	38	x-symbol-language-access-alist .....	62
x-symbol-context-init-ignore .....	29	x-symbol-language-value .....	62
x-symbol-copy-region-encoded .....	19	x-symbol-latin-force-use .....	13
x-symbol-decode .....	19	x-symbol-latin1-cset .....	52
x-symbol-decode-recode .....	19	x-symbol-latin1-fonts .....	12
x-symbol-default-coding .....	16	x-symbol-latin2-cset .....	52
x-symbol-default-context-info-ignore .....	38	x-symbol-latin2-fonts .....	12
x-symbol-electric-ignore .....	30	x-symbol-latin3-cset .....	52
x-symbol-electric-input .....	30	x-symbol-latin3-fonts .....	12
x-symbol-encode .....	19	x-symbol-latin5-cset .....	52
x-symbol-encode-recode .....	19	x-symbol-latin5-fonts .....	12
x-symbol-font-sizes .....	12	x-symbol-latin9-fonts .....	12
x-symbol-fontify .....	22, 65	x-symbol-list-bury .....	27
x-symbol-grid .....	27	x-symbol-list-info .....	27
x-symbol-grid-ignore-charsyms .....	28	x-symbol-list-mode-hook .....	31
x-symbol-grid-reuse .....	28	x-symbol-local-grid .....	27
x-symbol-grid-tab-width .....	28	x-symbol-local-menu .....	26
x-symbol-group-input-alist .....	31, 55	x-symbol-map-default-keys-alist .....	29
x-symbol-group-syntax-alist .....	62	x-symbol-menu-max-items .....	27
x-symbol-header-groups-alist .....	31	x-symbol-mode .....	21
x-symbol-heading-strut-glyph .....	28	x-symbol-modeline-state-list .....	21
x-symbol-help .....	28	x-symbol-modify-aspects-alist .....	54
x-symbol-idle-delay .....	33, 38	x-symbol-modify-key .....	29
x-symbol-image .....	34	x-symbol-mule-change-default-face .....	13
x-symbol-image-cache-directories .....	36	x-symbol-nomule-fontify-cstrings .....	22
x-symbol-image-colormap-allocation .....	35	x-symbol-package-bug .....	39
x-symbol-image-convert-colormap .....	34	x-symbol-package-info .....	38
x-symbol-image-convert-file-alist .....	35	x-symbol-package-url .....	6, 39
x-symbol-image-convert-mono-regexp .....	35	x-symbol-package-web .....	38
x-symbol-image-convert-program .....	7, 35	x-symbol-read-token .....	26
x-symbol-image-converter .....	7, 35	x-symbol-read-token-direct .....	26
x-symbol-image-current-marker .....	37	x-symbol-register-language .....	61
x-symbol-image-data-directory .....	37	x-symbol-reveal-invisible .....	33
x-symbol-image-editor .....	37	x-symbol-revealed-face .....	33
x-symbol-image-editor-alist .....	37	x-symbol-rotate-aspects-alist .....	54
x-symbol-image-max-height .....	34	x-symbol-rotate-key .....	29
x-symbol-image-max-width .....	34	x-symbol-rotate-prefix-alist .....	29
x-symbol-image-parse-buffer .....	36	x-symbol-rotate-suffix-char .....	29
x-symbol-image-scale-method .....	37	x-symbol-sgml-auto-coding-alist .....	47
x-symbol-image-searchpath-follow-symlink ..	34	x-symbol-sgml-auto-style .....	47
x-symbol-image-special-glyphs .....	37	x-symbol-sgml-class-alist .....	48
x-symbol-image-start-convert-color .....	35	x-symbol-sgml-class-face-alist .....	48
x-symbol-image-start-convert-colormap .....	35	x-symbol-sgml-electric-ignore .....	48
x-symbol-image-start-convert-mono .....	35	x-symbol-sgml-extra-menu-items .....	48
x-symbol-image-start-convert-truecolor .....	35	x-symbol-sgml-font-lock-alist .....	48

x-symbol-sgml-font-lock-contents-regexp ....	48	x-symbol-tex-font-lock-limit-regexp.....	42
x-symbol-sgml-font-lock-limit-regexp.....	48	x-symbol-tex-header-groups-alist.....	42
x-symbol-sgml-font-lock-regexp.....	48	x-symbol-tex-image-cached-dirs.....	43
x-symbol-sgml-header-groups-alist.....	48	x-symbol-tex-image-keywords.....	43
x-symbol-sgml-image-cached-dirs.....	48	x-symbol-tex-image-searchpath.....	43
x-symbol-sgml-image-file-truename-alist ....	48	x-symbol-tex-master-directory.....	43
x-symbol-sgml-image-keywords.....	48	x-symbol-tex-modes.....	41
x-symbol-sgml-image-searchpath.....	48	x-symbol-tex-preview-locations.....	8
x-symbol-sgml-master-directory.....	48	x-symbol-tex-token-suppress-space.....	42
x-symbol-sgml-modes.....	47	x-symbol-tex-user-table.....	46, 49
x-symbol-sgml-token-list.....	49	x-symbol-tex-verb-delimiter-regexp.....	45
x-symbol-sgml-token-list-code.....	49	x-symbol-texi-auto-style.....	50
x-symbol-sgml-token-list-name.....	49	x-symbol-texi-class-alist.....	50
x-symbol-sgml-token-list-netscape.....	49	x-symbol-texi-class-face-alist.....	50
x-symbol-sgml-user-table.....	49	x-symbol-texi-electric-ignore.....	50
x-symbol-subscripts.....	33	x-symbol-texi-extra-menu-items.....	50
x-symbol-temp-grid.....	27	x-symbol-texi-header-groups-alist.....	50
x-symbol-temp-help.....	29	x-symbol-texi-modes.....	50
x-symbol-tex-auto-coding-alist.....	41	x-symbol-texi-user-table.....	50
x-symbol-tex-auto-style.....	41	x-symbol-token-input.....	26
x-symbol-tex-class-alist.....	42	x-symbol-translate-to-ascii.....	38
x-symbol-tex-class-face-alist.....	42	x-symbol-unalias.....	20
x-symbol-tex-coding-master.....	41	x-symbol-unique.....	18
x-symbol-tex-electric-ignore.....	42	x-symbol-user-table.....	31
x-symbol-tex-electric-ignore-regexp.....	42	x-symbol-valid-charsym-function.....	25
x-symbol-tex-env-tabbing-regexp.....	46	x-symbol-xsymb0-cset.....	52
x-symbol-tex-env-verbatim-regexp.....	45	x-symbol-xsymb0-fonts.....	12
x-symbol-tex-error-location.....	8	x-symbol-xsymb1-cset.....	52
x-symbol-tex-extra-menu-items.....	42	x-symbol-xsymb1-fonts.....	12
x-symbol-tex-font-lock-allowed-faces.....	42	x-symbol-yank-decoded.....	20



## Concept Index

•  
 ‘.emacs’ ..... 6

### 8

8bit Character Problems ..... 67  
 8bit Coding Control ..... 17  
 8bit File Coding ..... 16

## A

Abbrev Problems ..... 63  
 Abbrev, Token ..... 26  
 About ..... 2  
 Accessing Language Depending Variables ..... 61  
 Acknowledgments ..... 80  
 Adding Fonts ..... 56  
 Additional Spaces ..... 66  
 Adobe ..... 80  
 Aggressive Context ..... 30  
 Aliases of Characters ..... 20  
 Allowed Character ..... 25  
 Alternative Auto Conversion ..... 61  
 Alternative Global Mode ..... 60  
 Alternative Token Representations ..... 60  
 AmsTeX ..... 77  
 Annoyances ..... 63  
 Annoying Subscripts ..... 66  
 Ascii Representation ..... 38  
 Ascii Sequence Input ..... 29  
 Aspects of Characters ..... 53  
 Auto Conversion, Alternatives ..... 61  
 Auto Initialization ..... 79  
 Automatic Context ..... 30  
 Automatic Conversion ..... 19  
 Avoiding Flickering ..... 59

## B

Basic Installation ..... 5  
 Basics ..... 15  
 Basics SGML Entity ..... 47  
 Basics TeX Macro ..... 41  
 bib ..... 49  
 BibTeX ..... 49  
 BibTeX macro ..... 49  
 Big Characters ..... 66  
 Binary Distribution ..... 5  
 Binary Package ..... 5  
 Bradfield, Julien ..... 80  
 Brief Summary ..... 1  
 Buffer Printing ..... 79  
 Bug Reports ..... 68  
 Bugs ..... 63  
 Built-in Languages ..... 41

## C

Caching of Images ..... 35  
 Category of Character ..... 23  
 Changes ..... 73  
 Changes in Emacs ..... 78  
 Changes in LaTeX ..... 78  
 Changes in XEmacs ..... 78  
 Char Aliases ..... 20  
 Character Aliases ..... 20  
 Character Descriptions ..... 53  
 Character Descriptions, Example ..... 53, 54  
 Character Descriptions, Intro ..... 53  
 Character Group ..... 23  
 Character Info ..... 37  
 Character Insertion ..... 25  
 Character Problems ..... 65  
 Character Sequence Input ..... 29  
 Character Terminal ..... 5  
 Charset ..... 51  
 Charsym ..... 41, 51  
 Checking Installation ..... 13  
 Choosing SGML Entity ..... 47  
 Choosing TeX Macro ..... 41  
 Classes of Tokens ..... 23  
 Coding in File ..... 16  
 Coding, Default ..... 15  
 Coloring Scheme ..... 23  
 Colormap ..... 34  
 Component of Characters ..... 53  
 Compose Key ..... 25  
 Compression Packages ..... 9  
 Concepts ..... 15  
 Consistent Input Methods ..... 52  
 Contacting the Maintainer ..... 68  
 Context Info ..... 37  
 Context, Input Method ..... 29  
 Contributions ..... 77, 80  
 Controlling 8bit Coding ..... 17  
 Controlling Images ..... 34  
 Conversion ..... 15  
 Conversion Commands ..... 19  
 Conversion of SGML Entities ..... 49  
 Conversion of TeX Macros ..... 45  
 Conversion Problems ..... 66  
 convert Installation ..... 7  
 Converting Images ..... 34  
 Copy and Conversion ..... 19  
 Copy Encoded ..... 19  
 Copying ..... 1  
 Copyright ..... 1  
 Core XEmacs ..... 64  
 Crash XEmacs ..... 64  
 Cset ..... 51  
 Cstring ..... 51  
 Cursor, Invisible ..... 33  
 Customizing Input Methods ..... 55  
 Customizing Method Internals ..... 31

**D**

Decoding .....	15
Default Coding .....	15
Default Encoding .....	15
Default Font .....	15
Default Languages .....	41
' <code>default.el</code> ' .....	6
Defined Character .....	25
Defining Input Methods .....	52
Defining <code>tex</code> .....	46
Designing Images .....	37
Documentation .....	13

**E**

Echo Area Info .....	37
Editing Image Files .....	37
Electric, Input Method .....	30
Elisp Installation .....	6
Emacs Changes .....	78
Email to the Maintainer .....	68
Encoding .....	15
Encoding in File .....	16
Encoding Problems .....	66
Encoding, Default .....	15
Encryption Packages .....	9
Escape Character Problems .....	65
Exclusive Modify Chain .....	53
Explicit Conversion .....	19
Extending with Fonts .....	56
Extending X-Symbol .....	55
External Languages .....	50
Extra Symbols for <code>TeX</code> .....	46
Extract Tarball .....	5

**F**

FAQ X-Symbol .....	64
Features of X-Symbol .....	33
Features SGML Entity .....	48
Features <code>TeX</code> Macro .....	42
File Cache for Images .....	35
File Coding .....	16
File I/O Packages .....	9
Fill Problems .....	63
Final Byte .....	51
Final Installation Checks .....	13
Flickering, Invisible .....	59
Font Definition File .....	57
Font Extension .....	56
Font Lisp Installation .....	12
Font Lisp Setup .....	12
Font Size .....	66
Font, Default .....	15
<code>font-lock</code> Packages .....	8
<code>font-lock</code> Problems .....	65
<code>font-lock</code> Use .....	22
Fonts .....	10, 11
Fonts for Windows .....	77
Fonts from Other Sources .....	77
Foreign Languages .....	50
Frequently Asked Questions .....	64
Funny Characters .....	65
Future Features .....	77

**G**

General Public License .....	1
Generated Fonts .....	77
Global Mode, Alternatives .....	60
Glyph Caching .....	35
Glyph for Specific Situations .....	36
Glyphs .....	33
GPL .....	1
Greek Input .....	29
Grid, Input Method .....	27
Group of Characters .....	23
Guidelines, Font Extension .....	56
Guidelines, Input Definitions .....	56

**H**

Highlighted Character .....	27
History .....	73
Horizontal Chain .....	53
HTML .....	13, 47
Hyphen Versus Minus .....	67

**I**

I/O Packages .....	9
Image Cache File .....	34
Image Caching .....	35
Image Commands .....	34
Image Control .....	34
Image Conversion .....	34
Image Converter Installation .....	7
Image Display .....	34
Image Editor .....	37
Image Highlight Menu .....	37
Image Keywords .....	34
Images .....	33
Images for Specific Situations .....	36
Info .....	13
Info in Echo Area .....	37
Info Pages .....	2
Initialized Language .....	61
Initializing <code>tex</code> .....	46
Input Definitions, Guidelines .....	56
Input Method Context .....	29
Input Method Electric .....	30
Input Method Grid .....	27
Input Method Internals .....	52
Input Method Keyboard .....	28
Input Method Menu .....	26
Input Method Objectives .....	52
Input Method Quail .....	30
Input Method Read Token .....	26
Input Method Token .....	26, 64, 79
Input Methods .....	25
Input Methods Customization .....	31, 55
Input Methods, Common .....	25
Input Methods, General .....	25
Input Methods, Standard .....	25
' <code>inputenc.sty</code> ' Changes .....	78
Insert Commands, Tagging .....	59
Installation .....	5
Installation Checks .....	13
Installing <code>convert</code> .....	7
Installing Fonts .....	10, 11

Installing Image Converter .....	7
Installing Lisp .....	6
Installing Manual .....	13
Installing System-wide .....	6
Installing <b>tex</b> .....	46
Integrating <b>font-lock</b> Packages .....	8
Integrating I/O Packages .....	9
Integrating LaTeX Packages .....	8
Integrating Package .....	7
Interactive Conversion .....	19
Internals, Input Method .....	52
Internals, Languages .....	61
Internals, X-Symbol .....	51
Introduction .....	1
Invisible Flickering .....	59
Invisible Point .....	63
Invisible, Revealing .....	33
IPA Fonts .....	77
Isabelle Symbol .....	50

## K

Key Chain .....	53
Key Prefix .....	25
Keyboard, Input Method .....	28
Keywords for Images .....	34
Keywords for Subscripts .....	33

## L

Label Creation .....	38
Label Subscripts .....	66
Language .....	15
Language Access .....	61
Language Additions .....	77
Language <b>bib</b> .....	49
Language Definition File .....	57, 58
Language Internals .....	61
Language <b>sgml</b> .....	47
Language <b>tex</b> .....	41
Language <b>texi</b> .....	50
Languages in Distribution .....	41
LaTeX .....	41
LaTeX Changes .....	78
LaTeX Packages .....	8
Latin Character Aliases .....	20
Latin File Coding .....	16
Latin in File .....	17
Leading Character .....	51
License .....	1
Lisp Installation .....	6
Loaded Language .....	61

## M

Mail Reader .....	68
Maintainer Address .....	68
Manual .....	2, 13
Marlet, Renaud .....	80
Memory Cache for Images .....	35
Menu, Input Method .....	26
Minibuffer Completion, Token .....	26
Minibuffer Info .....	37
Minor Mode .....	20

Minus Versus Hyphen .....	67
Miscellaneous Packages .....	10
Mode .....	20
Modify Aspects .....	53
Modify Chain .....	29, 53
Modify Scores .....	53
More Token Languages .....	77
MS-Windows .....	5
MS-Windows Fonts .....	77
Mule Character .....	51
Mule Input Method .....	30

## N

New Features .....	73
News Reader .....	68
No 8bit Characters .....	67
No Encoding .....	64
No fontification .....	65
No Mule .....	22
No Subscripts .....	65
Nomule Problems .....	63

## O

Objectives, Input Methods .....	52
Octet .....	51
Old Version .....	5
Online Help .....	2
Open Questions .....	79
Other Languages .....	50
Other Packages .....	7
Overview .....	1

## P

Package Information .....	38
Package Integration .....	7
Parent Character .....	53
Parenthesis Problems .....	63
Paste and Conversion .....	19
Paste Decoded .....	19
Point, Invisible .....	33
Poor Man's Mule .....	22
Postscript .....	13
Prefix Argument .....	25
Prefix Key .....	25
Print Buffer .....	79
Problem Reports .....	68
Problems .....	63
Problems TeX Macro .....	43
Project .....	77
Provided Languages .....	41
Pseudo Language .....	41

## Q

Questions I Have .....	79
------------------------	----

**R**

Read Token, Input Method .....	26
Recoding .....	16
Recognizing Insert Commands .....	59
Rectangle Problems .....	63
Registered Languages .....	61
Rejected Suggestions .....	79
Remapping Characters .....	20
Remote File Packages .....	9
Replace Problems .....	63
Replace Token .....	26
Reports of Bugs .....	68
Representation of Characters .....	38
Requirements .....	5
Restricted Decoding .....	18
Revert Buffer Problems .....	63
Role of <code>font-lock</code> .....	22
Rotate Aspects .....	53
Rotate Chain .....	29, 53
Rotate Scores .....	53

**S**

Scale Factor, Images .....	37
Score of a Character .....	53
SGML entity .....	47
SGML Entity Basics .....	47
SGML Entity Conversion .....	49
SGML Entity Features .....	48
SGML Entity Images .....	48
SGML Entity Modes .....	47
SGML Entity Subscripts .....	48
SGML Entity Superscripts .....	48
SGML Entity Use .....	47
Similar Characters .....	23
' <code>site-start.el</code> ' .....	6
Small Characters .....	66
Source Package .....	5
Space Problems .....	66
Special Fonts .....	22
Special Images .....	36
Spell Checking .....	67
Spurious Encodings .....	63
Storing 8bit Characters .....	17
Strange Characters .....	65
Stupid Subscripts .....	66
Subscript Problems .....	65, 66
Subscripts .....	33
Summary .....	1
Superscript Problems .....	65, 66
Superscripts .....	33
Supported Features .....	33
Supported Languages .....	41
Syntax Highlighting Packages .....	8
Syntax of Character .....	23
System-wide Installation .....	6

**T**

Table of Characters .....	27
Tagging Insert Commands .....	59
Tarball .....	5
Terminology .....	15
<code>tex</code> .....	41

TeX Macro Basics .....	41
TeX Macro Conversion .....	45
TeX Macro Features .....	42
TeX Macro Images .....	42
TeX Macro Modes .....	41
TeX Macro Problems .....	43
TeX Macro Subscripts .....	42
TeX Macro Superscripts .....	42
TeX Macro Use .....	41
<code>TeX</code> macro .....	41
TeX Macro Installation .....	46
<code>TeXinfo</code> command .....	50
<code>texi</code> .....	50
Texinfo .....	13
Thanks .....	80
Token Changes .....	66
Token Classes .....	23
Token Language .....	15
Token Language <code>bib</code> .....	49
Token Language Internals .....	61
Token Language <code>sgml</code> .....	47
Token Language <code>tex</code> .....	41
Token Language <code>texi</code> .....	50
Token Problems .....	66
Token Representation, Alternatives .....	60
Token, Input Method .....	26
<code>transpose-chars</code> Problems .....	63
Troubleshooting .....	63
TTY .....	5
Turn on Globally, Alternatives .....	60

**U**

Uncompress Tarball .....	5
Unique Decoding .....	18
Unique TeX macro .....	18
URL for X-Symbol .....	38

**V**

Valid Character .....	25
Various Questions .....	79
<code>vc</code> Problems .....	63
Version Control Problems .....	63

**W**

Warranty .....	1
Web Pages .....	2
Window System .....	5
Windows .....	5, 11
Windows Fonts .....	77
Wishlist .....	77
<code>write-file</code> Problems .....	63
<code>write-region</code> Problems .....	64
WWW Browsing .....	38

**X**

X .....	5
X-Server .....	11
X-Symbol Charsym .....	41
X-Symbol in a Nutshell .....	1
X-Symbol Internals .....	51
X-Symbol Mode .....	20
X11 Fonts .....	10

XEmacs Changes .....	78
XEmacs Core .....	64
XEmacs without Mule .....	22

**Y**

Yank Decoded .....	19
Your Contribution .....	77

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	X-Symbol's Copying Conditions: GPL .....	1
1.2	Brief Summary of X-Symbol .....	1
1.3	About this Manual .....	2
<b>2</b>	<b>Installation .....</b>	<b>5</b>
2.1	Requirements .....	5
2.2	Put the Files into your Home Directory .....	5
2.3	System-wide Installation: Put the Files into the XEmacs Directory .....	6
2.4	Make XEmacs Initialize X-Symbol During Startup .....	6
2.5	Installing the Image Converter from ImageMagick .....	7
2.6	Package Integration .....	7
2.6.1	LaTeX Packages .....	7
2.6.2	Syntax Highlighting Packages ( <code>font-lock</code> and add-ons) .....	8
2.6.3	File I/O Packages .....	9
2.6.4	Miscellaneous Packages .....	10
2.7	Installing Additional Fonts .....	10
2.8	Installing Fonts for Exceed (X-server on Windows) .....	11
2.9	Lisp Coding when Using Other Fonts .....	12
2.10	Installing Info, Postscript and HTML Files .....	13
2.11	Checking the Correct Installation of Package X-Symbol .....	13
<b>3</b>	<b>Concepts of Package X-Symbol .....</b>	<b>15</b>
3.1	Token Language .....	15
3.2	Conversion: Decoding and Encoding .....	15
3.2.1	Normal File and Default Encoding .....	15
3.2.2	File Coding of 8bit Characters .....	16
3.2.3	Store or Encode 8bit Characters .....	17
3.2.4	Unique Decoding .....	18
3.2.5	Conversion Commands .....	18
3.2.6	Copy & Paste with Conversion .....	19
3.2.7	Character Aliases .....	20
3.3	Minor Mode .....	20
3.4	Poor Man's Mule: Running Under XEmacs/no-Mule .....	22
3.5	The Role of <code>font-lock</code> .....	22
3.6	Character Group and Token Classes .....	22
<b>4</b>	<b>X-Symbol's Input Methods .....</b>	<b>25</b>
4.1	Common Behavior of All Input Methods .....	25
4.2	Input Method Token: Replace Token by Character .....	25
4.3	Input Method Read Token: Minibuffer Completion .....	26
4.4	Input Method Menu: Select a Menu Item .....	26
4.5	Input Method Grid: Choose Highlighted Character .....	27
4.6	Input Method Keyboard: Compose Key Sequence .....	28
4.7	Input Method Context: Replace Char Sequence .....	29
4.8	Input Method Electric: Automatic Context .....	29
4.9	Input Method Quail: a Mule Input Method .....	30
4.10	Customizing Input Methods .....	30

<b>5</b>	<b>Features of Package X-Symbol</b>	<b>33</b>
5.1	Super- and Subscripts	33
5.2	Images at the end of Image Insertion Commands	33
5.2.1	Display of Images	33
5.2.2	Image Conversion	34
5.2.3	Image Caching	35
5.2.4	Special Images for Specific Situations	36
5.2.5	Image Editor	37
5.3	Info in Echo Area	37
5.4	Ascii Representation of Strings	38
5.5	X-Symbol Package Information	38
<b>6</b>	<b>Supported Token Languages</b>	<b>41</b>
6.1	Pseudo Token Language “x-symbol charsym”	41
6.2	Token Language “ $\TeX$ macro” ( <code>tex</code> )	41
6.2.1	Basics of Language “ $\TeX$ macro”	41
6.2.2	Super-/Subscripts and Images in $\LaTeX$	42
6.2.3	Problems with $\TeX$ Macros	43
6.2.4	The Conversion of $\TeX$ Macros	44
6.2.5	Extra Symbols of Language “ $\TeX$ Macro”	46
6.3	Token Language “SGML entity” ( <code>sgml</code> )	47
6.3.1	Basics of Language “SGML entity”	47
6.3.2	Super-/Subscripts and Images in HTML	48
6.3.3	The Conversion of SGML Entities	48
6.4	Token Language “Bib $\TeX$ macro” ( <code>bib</code> )	49
6.5	Token Language “ $\TeX$ info command” ( <code>texi</code> )	49
6.6	Languages Defined in Other Emacs Packages	50
<b>7</b>	<b>X-Symbol Internals</b>	<b>51</b>
7.1	Internal Representation of X-Symbol Characters	51
7.2	Defining X-Symbol Charsets	51
7.3	Defining Input Methods	52
7.3.1	Defining Input Methods: Objectives	52
7.3.2	X-Symbol Character Descriptions: Example	53
7.3.3	Defining Input Methods by Character Descriptions	53
7.3.4	Defining Input Methods: Example	54
7.3.5	Customizing Input Methods	55
7.4	Extending Package X-Symbol	55
7.4.1	Extending X-Symbol with New Fonts	55
7.4.2	Guidelines for Input Definitions	56
7.4.3	Emacs Lisp File Defining a New Font	57
7.4.4	Emacs Lisp File Extending a Token Language	57
7.4.5	Emacs Lisp File Defining a New Token Language	58
7.5	Various Internals	59
7.5.1	Tagging Insert Commands for Token and Electric	59
7.5.2	Avoiding Hide/Show-Invisible Flickering	59
7.6	Design Alternatives	59
7.6.1	Alternative Token Representations	59
7.6.2	Alternative Ways to Turn on X-Symbol Globally	60
7.6.3	Alternative Auto Conversion Methods	61
7.7	Language Internals	61
7.8	Miscellaneous Internals	62



<b>8</b>	<b>Problems, Troubleshooting</b>	<b>63</b>
8.1	Problems under XEmacs/no-Mule	63
8.2	Spurious Encodings	63
8.3	The Encoding Does Not Work	64
8.4	Frequently Asked Questions	64
8.4.1	XEmacs Crashes when using Input Method Token	64
8.4.2	X-Symbol's Fontification does Not Work	64
8.4.3	The Buffer Contains Strange Characters	65
8.4.4	I Cannot See any/some Super- or Subscripts	65
8.4.5	I See Super- and Subscripts where I Don't Want Them	66
8.4.6	The Characters are Too Small or Too Big	66
8.4.7	The Conversion Changes Some Tokens	66
8.4.8	A Space is Added During the Encoding	66
8.4.9	I Don't Want 8bit Characters in the File	67
8.4.10	I Cannot Distinguish Character <code>hyphen</code> from <code>'-'</code>	67
8.4.11	Problems with Spell-checking	67
8.4.12	How to Use X-Symbol with Gnus or VM	67
8.5	How to Send a Bug/Problem Report	68
<b>9</b>	<b>History and Projects</b>	<b>73</b>
9.1	News: Changes in Recent Versions of X-Symbol	73
9.1.1	Changes in X-Symbol 4.5.1	73
9.1.2	Changes in X-Symbol 4.5	73
9.1.3	Changes in X-Symbol 4.2 to 4.4	73
9.1.4	Changes in X-Symbol 4.1	74
9.1.5	Changes in X-Symbol 3.4	74
9.1.6	Changes in X-Symbol 3.3	74
9.1.7	Changes in X-Symbol 3.2	75
9.1.8	Changes in X-Symbol 3.1	75
9.1.9	Changes in X-Symbol 3.0	75
9.1.10	Changes in Old Releases	76
9.2	Wishlist: Projects for X-Symbol	77
9.2.1	Wishlist: Additional Token Languages	77
9.2.2	Wishlist: Generated Fonts	77
9.2.3	Wishlist: Changes in Emacs/XEmacs	78
9.2.4	Wishlist: Changes in LaTeX	78
9.2.5	Various Projects for X-Symbol	79
9.2.6	Rejected Suggestions for X-Symbol	79
9.3	Open Questions	79
9.4	Acknowledgments	79
	<b>Indexes</b>	<b>83</b>
	Key Index	83
	Program and Package Index	84
	Command, Function and Variable Index	85
	Concept Index	88

