**Abstract**

Eplain is a macro package for the TEX typesetting engine. This document describes implementation of Eplain's hypertext link support. For usage instructions refer to Eplain user manual [7].

Oleg Katsitadze

April 2006

## Table of Contents

# 1 Background

TEX is a typesetting system developed by Donald E. Knuth [15]. TEX works by processing an input '`.tex`' file which contains text interspersed with commands, such as commands to change the typeface, insert space, add accents, etc.

The low-level, atomic commands provided by TEX are called *primitives*. Considered individually, primitives provide very limited and specialized functionality. Sequences of primitives (again, possibly mingled with text) are usually combined together in a *macro*. Therefore macros provide more powerful typesetting capabilities than individual primitives. Macros can in turn be combined to any level of nesting to form other macros.

Such intricate system of macros can be collected in a file or several files to form a *macro package*, or a *format file*. Some macro packages specialize on particular features, such as fonts and typefaces, typesetting of music or lyrics. Other macro packages provide general capabilities useful in development of other macros and packages.

As part of his work on TEX, Donald E. Knuth developed plain TEX [14], a general-purpose macro package. It was followed by many other macro packages: Eplain [7], $\mathcal{AMS}$-TEX [5, 22], LATEX [16, 17], Texinfo [9], ConTEXt [19], Lollipop [10] and many others. Nowadays LATEX is by far the most popular macro package.

## 1.1 Eplain

The original plain TEX is more or less a low-level interface to TEX primitives (while providing a number of user-oriented macros). It lacks many features which most document writers will reasonably expect, implementation of which requires some experience in TEX macro programming language. A good example is using labels for cross-references. Instead of manually inserting absolute numbers throughout the manuscript, authors like to assign labels to various parts of the document such as sections, figures, etc. When later TEX encounters a cross-reference with a label, it automatically changes the label to the appropriate number. This saves lots of manual work when parts of the document are reordered.

The Eplain macro package was written by Karl Berry as part of the book *TEX for the Impatient* [1], published by Addison-Wesley in 1990. Eplain expands on and extends the definitions in plain TEX, providing features such as symbolic cross-referencing, lists, citations, indexing and many other capabilities. Eplain provides both macros intended to be used directly in documents and macros to be used as tools in developing formats.

### Eplain vs. LATEX

As LATEX is the most popular macro package for TEX, a note on how Eplain compares to LATEX is in place.

The philosophy of Eplain is to provide functionality which can be used (or not used) as desired, without forcing any typographic style on an author. This approach is much different from that of LATEX. LATEX hides many details from the user, making it easier to write documents with predefined styles and harder to produce bad typesetting. A vast number of additional packages has been developed to allow users to customize LATEX, but desired changes can still sometimes be hard to accomplish.

LaTeX is sometimes regarded as an easier-to-learn package, suitable for beginners and occasional TeX users; however, many TeX experts are LaTeX users, of which many participate in TeX and LaTeX development. On the other hand, plain TeX and many plain-TeX-based packages (of which Eplain is one) are regarded as "hacker-intensive", and are mostly used by professional compositors and designers or people who know TeX internals well and like to keep TeX under complete control.

## 1.2 TeX and electronic document interchange

While TeX documents can be distributed in the form of source '`.tex`' files, this requires recipients to have a working installation of TeX in order to process the '`.tex`' files prior to viewing or printing. Another complication is that to produce identical outputs, systems must have fonts with identical metrics installed, which is cumbersome and not always possible due to technical or legal issues.

TeX's native output format, DVI (from "**de**vice **i**ndependent") is still not appropriate for document interchange. Most importantly, it does not allow embedding of actual glyphs of fonts used by the document, it can only store general information about those fonts, so the problem of availability of compatible font files persists.

All these problems can be overcome with Adobe's PostScript format and Portable Document Format (PDF). Both of these formats allow font embedding and preserve exactly layout of a document. Several utilities had been created for conversion from DVI to PostScript [20], from DVI to PDF [13, 24] and from PostScript to PDF [11], as well as a TeX engine capable of outputting PDF directly [23, 25].

PostScript [2] is a page description language designed to convey a description of a page to a printer. While it was intended as a language to be "understood" by printer drivers and even directly by some high-end printers, software packages had been developed which allow viewing PostScript documents on a computer monitor [11]. PostScript format has become an unofficial standard among professionals, but remains exotic among unsophisticated users, especially on Microsoft Windows.

On the other hand, Portable Document Format [3] was designed explicitly for electronic document interchange. As such, it has some features beneficial to electronic documents which are not available in PostScript, for example, hypertext links. Furthermore, PDF is commonplace on a wider variety of platforms, compared to PostScript.

This makes PDF the best candidate for distribution of electronic documents produced with TeX.

### Hypertext links in TeX

The original TeX engine has no built-in support for hypertext links (a.k.a. hyperlinks). Many of the present-day file formats with hyperlinking capabilities did not even exist at the time TeX was written. However, TeX's `\special` primitive can be used to instruct TeX to write special directives into its DVI output file. These directives are not interpreted by TeX in any way; they are intended for programs which process the DVI files, be it printing or converting to other formats, such as PostScript or PDF. This approach is used by DVI-to-PDF and DVI-

to-PostScript-to-PDF[1] converters (such as `dvipdfmx` [13], and `dvips` [20] with Ghostscript [11]) to embed special features of the PDF format.

For example, invoking this `\special` command from TEX:

```
\special{pdf: dest (⟨label⟩) [@thispage /FitH @ypos]}
```

writes into the output DVI file a directive which instructs `dvipdfmx` to create a PDF hyperlink destination (`dest`) named ⟨*label*⟩ on the current page (`@thispage`) which fits the width of the page to the viewer window (`/FitH`) at the current vertical position (`@ypos`).

Another approach is to extend the original TEX engine with the ability to generate one of the hyperlink formats—TEX's set of primitives can be extended to include hyperlink commands. This is the approach used by the pdfTEX engine [23, 25] which is capable of producing PDF files directly from TEX sources, skipping the DVI generation and processing step.

For example, this is pdfTEX equivalent of the above `\special` command:

```
\pdfdest name{⟨label⟩} fith
```

### Hypertext macros

In principle, it is possible to create a document with hypertext links with TEX by directly specifying `\special` commands for a DVI processor or extended pdfTEX primitives in the document. However, this is inconvenient, and higher level macros can greatly simplify the use of hypertext links.

Several macro packages had been developed for this purpose. For example, the `hyperref` package for LATEX "extends the functionality of all the LATEX cross-referencing commands (including the table of contents, bibliographies, etc.) to produce `\special` commands which a driver can turn into hypertext links; it also provides new commands to allow the user to write *ad hoc* hypertext links, including those to external documents and URLs" [21]. Unfortunately, `hyperref` is tightly integrated with LATEX and cannot be used with plain TEX or packages based on plain TEX.

The `hyperbasics` [8] and `lanlmac` [12] packages provide hypertext functionality for plain TEX. These packages are based on HyperTEX [18], a standard for embedding hypertext links in the DVI file. This standard is by necessity quite limited and cannot take advantage of many hypertext-related features of PDF.

### Hypertext links in Eplain

This document describes an attempt to provide hypertext capabilities to plain TEX users through the Eplain macro package, similar to what `hyperref` provides to LATEX users. Besides providing simple wrappers around low-level `\special` commands or primitives, existing Eplain macros are extended to automatically create hypertext links for cross-references, citations, indexes, etc.

The following chapters describe implementation of the hyperlink support in Eplain; for usage instructions refer to Eplain user manual [7].

Hypertext link support was introduced in Eplain starting with version 3.0 [26].

[1] Although PostScript does not support hyperlinks, Adobe introduced the `pdfmark` operator [4] as a PostScript language extension, which can be used to describe features that are present in PDF but not in standard PostScript. The use of the `pdfmark` operator requires support by PostScript-to-PDF conversion software. Adobe Acrobat Distiller and Ghostscript [11] support `pdfmark`.

## 2 Implementation overview

In the following discussion, we will use the term *link* to refer to a hypertext link, and the term *destination* to refer to a hypertext destination (a.k.a. target or anchor). When referring to links and destinations jointly, we will use the term *hyperlinks*. Line numbers in brackets in typewriter type (like this: `[1-10]`) refer to the code listing 8.1 (see Appendix, p. 17).

As mentioned above, several options exist for producing PDFs with hypertext links with TEX. To provide a way to support all these possibilities as well as future developments, hyperlink support is structured to separate generic hyperlink macros independent of a PDF engine used, from macros defining functionality specific to a PDF engine. These latter macros are organized as *hyperlink drivers* (see Chapter 3, p. 7).

The generic hyperlink macros can be used directly by the user to create *explicit* hyperlinks (see Chapter 4, p. 9). For example,

```
\hlstart{url}{}{http://tug.org}
  \TeX{} Users Group web site\hlend
```

typesets the text "TEX Users Group web site" as an explicit link to the specified URL, `http://tug.org`.

These generic hyperlink macros are also used internally by several Eplain macros to create *implicit* hyperlinks. For example, referring to a defined cross-reference with the command

```
\ref{Chapter:Implementation}
```

converts the chapter reference into a hyperlink behind the scenes. Eplain macros which implicitly generate destinations are assigned to one of the *destination groups* (or *destgroups* for short); Eplain macros which implicitly generate links are assigned to one of the *link groups* (or *linkgroups* for short) (see Chapter 5, p. 11).

Several macros are provided to control various aspects of hyperlinks, such as type, width and color of link border, hypertext color, etc. These macros are described in Chapter 6, p. 12.

It is possible to temporarily or permanently disable or enable all hyperlinks at once or any hyperlink group selectively. The macros to do this are described in Chapter 7, p. 14.

# 3 Hyperlink drivers

Version 3.0 of Eplain was released with two "real" drivers, `pdftex` and `dvipdfm`, and one pseudo-driver, `nolinks`. As of this writing (April 2006), development version of Eplain also provides the `hypertex` driver.

## 3.1 Defining a driver

In this section we will look at how a new hyperlink driver can be defined.

When the user calls `\enablehyperlinks[`⟨*driver_name*⟩`]` [552-628], the presence of the driver is established by checking that a macro with the name `\hldriver@`⟨*driver_name*⟩ is defined [600]. If `\enablehyperlinks` finds the requested driver, it calls the command `\hldriver@`⟨*driver_name*⟩, which is expected to define several commands with special names (see Table 3.1). These special commands are what defines the behavior of the driver—they are used by the generic hyperlink macros (see Chapter 4, p. 9) to handle driver-specific part of hyperlink creation.

TABLE 3.1

Commands which should be defined by a hyperlink driver

| *Command name* | *Description* |
|---|---|
| `\hldest@driver` | This is a multiplexer for all supported destination types. The type is passed in `\@hltype`; the label is passed in `\@hllabel`. |
| `\hldest@type` | Name of the default destination type, to be used when no type is specified. |
| `\hldest@typeh@`⟨*name*⟩ | Should be defined for each destination type ⟨*name*⟩ the driver supports. Actual definition does not matter. |
| `\hldest@opt@`⟨*name*⟩ | Should be defined to a default value for each destination option ⟨*name*⟩ the driver supports (except `raise`, see Section 4.1, p. 9). These will be set according to user preferences before calling `\hldest@driver` (see Section 4.3, p. 9). The driver should consult current values of relevant options when creating destinations. |
| `\hl@driver` | This is a multiplexer for all supported link types. The type is passed in `\@hltype`; the label is passed in `\@hllabel`. |
| `\hl@type` | Name of the default link type, to be used when no type is specified. |
| `\hl@typeh@`⟨*name*⟩ | Should be defined for each link type ⟨*name*⟩ the driver supports. Actual definition does not matter. |
| `\hl@opt@`⟨*name*⟩ | Should be defined to a default value for each link option ⟨*name*⟩ the driver supports (except `colormodel` and `color`, see Section 4.2, p. 9). These will be set according to user preferences before calling `\hl@driver` (see Section 4.3, p. 9). The driver should consult current values of relevant options when creating links. |
| `\@hlend` | Should end the link started by `\hl@driver`. |

## 3.2 Driver `dvipdfm`

This driver [957-1154] implements the `\special` commands supported by the `dvipdfm` [24] and `dvipdfmx` [13] DVI-to-PDF converters. This driver allows exten-

sive customization of hyperlinks. See [7] for description of the types and options supported by this driver.

### 3.3 Driver `hypertex`

This driver [702-781] implements the HyperTEX standard [18]. This driver supports very limited customization due to the nature of the HyperTEX standard. See [7] for description of the types and options supported by this driver.

### 3.4 Driver `pdftex`

This driver [782-956] implements the extended primitives of the pdfTEX engine [23, 25]. This driver allows extensive customization of hyperlinks. See [7] for description of the types and options supported by this driver.

### 3.5 Pseudo-driver `nolinks`

Hyperlink macros use several features of TEX which can possibly affect spacing and page-breaking. The `nolinks` driver [629-701] is a pseudo-driver for situations when you've prepared a document with hyperlinks and just want to compile a version without them. This driver omits all hyperlinks in the document but ensures that spacing and page-breaking are the same as what you were getting with hyperlinks enabled. Details can be found in [7].

# 4 Explicit hyperlinks

In this chapter we'll look at the generic hyperlink macros which work on top of the hyperlink driver and provide the basis for user-level explicit hyperlink macros.

## 4.1 Destinations

`\@hldest` [5-20] is the macro which is called to create a destination. This is what ends up as `\hldest` when the user enables hyperlinks [544].

`\@hldest` works by first parsing its list of options (see Section 4.3) and then calling the driver (`\hldest@driver`, see Section 3.1, p. 7).

Unlike all other options which are handled by the driver, the option `raise` is handled in `\hldest@aftergetparam` [21-41]. This is because raising of destinations is handled the same way for all drivers using the standard TEX primitives. Destinations are raised by placing them inside a box of zero width, height and depth, and then raising that box to the requested height using TEX's `\raise` primitive. Raising is made only in horizontal mode; in vertical mode the destination is placed directly in the vertical list, without an enclosing box.

## 4.2 Links

`\@hlstart` [44-61] and `\@hlend` (see Section 3.1, p. 7) are the macros which are called to create a hyperlink. They end up as `\hlstart` and `\hlend`, respectively, when the user enables hyperlinks [536-539].

`\@hlstart` works by first parsing its list of options (see Section 4.3) and then calling the driver (`\hl@driver`, see Section 3.1, p. 7).

Unlike all other options which are handled by the driver, the two common options, `colormodel` and `color`, are handled in `\hlstart@aftergetparam` [61-76]. This is because coloring is not directly supported by the drivers. Instead, we rely on the user to define the `\color` command which should take care of the coloring (for example, by loading LATEX's `color` package which is supported by Eplain; see [7], section on using LATEX packages with Eplain). The calling sequence for the `\color` command is the following:

`\color[`⟨*color_model*⟩`]{`⟨*color*⟩`}`

where `[`⟨*color_model*⟩`]` is optional and can be omitted. The actual format of the ⟨*color_model*⟩ and ⟨*color*⟩ parameters is not in any way controlled by the hyperlink macros, therefore the user is responsible for setting the `colormodel` and `color` hyperlink options to appropriate values. Note, however, that default values for these options are:

```
colormodel=cmyk
color=0.28,1,1,0.35
```

If the user's `\color` command does not understand such specifications, the user might need to override the default values of the `colormodel` and `color` link options prior to using the link macros.

## 4.3 Handling of options

Parsing and setting of options is identical for `\@hldest` and `\@hlstart`, therefore it is done by the `\hl@getparam` macro [79-118] which is called by both `\@hldest`

and \@hlstart. The options are set locally within a TEX group, so that the setting is only effective for the current instance of the hyperlink macro.

\hl@getparam first checks that the requested destination or link type is supported by the driver, by checking that the command \hl@typeh@⟨*name*⟩ or \hldest@typeh@⟨*name*⟩ is defined [96] (see Section 3.1, p. 7).

Next, for each option in the comma-separated list of option assignments of the form ⟨*option*⟩=⟨*value*⟩, \hl@getparam calls \hl@set@opt [128-143] which ensures that the option is supported by the driver (by checking that the command \hl@opt@⟨*name*⟩ or \hldest@opt@⟨*name*⟩ is defined [130], see Section 3.1, p. 7) and then sets that option to the requested value.

Options not listed in the option list automatically retain previous values.

After parsing and setting all options from the option list, \hl@getparam calls \after@hl@getparam [123] (set to \hldest@aftergetparam by \@hldest [15] and to \hlstart@aftergetparam by \@hlstart [55]); \after@hl@getparam handles the driver-independent options (raise for destinations, colormodel and color for links, see Section 4.1, p. 9 and Section 4.2, p. 9) and then calls the driver to create destination or link using the options that have been set.

# 5 Implicit hyperlinks

Many Eplain macros create hyperlinks implicitly. They do so by using the macros `\hldest@impl [148-157]` (for destinations) and `\hlstart@impl [158-168]` and `\hlend@impl [169-174]` (for links). These macros take name of a linkgroup or destgroup and, if the specified group has not been disabled (see Section 7.2, p. 14), call `\hldest`, `\hlstart` or `\hlend`, passing the group's type and options (see Section 6.2, p. 12).

All destgroups should be listed in `\hldest@groups [187]`; all linkgroups should be listed in `\hl@groups [188]`. These are used by the macros which set types and options (see Chapter 6, p. 12) and turn hyperlinks on/off (see Chapter 7, p. 14) to process the special group '`*`' which expands to all defined groups.

# 6 Hyperlink types and options

Each driver declares link and destination types it supports and options which the user may set to control various aspects of hyperlinks (see Section 3.1, p. 7). Section 6.1 describes macros for setting default types and options which are used when the type or an option are not specified by the user in a call to an explicit hyperlink macro (see Section 4.3, p. 9) and when the type or an option are not defined for a hyperlink group (see Chapter 5, p. 11). Macros for setting hyperlink group types and options are described in Section 6.2.

`\hldesttype` [200–205], `\hldestopts` [206–211], `\hltype` [212–217] and `\hlopts` [218–223] are the top-level interface macros for the `\hl@setparam` macro which does the actual parsing of their parameters and setting of types and options. When hyperlinks are not enabled, `\hl@setparam` is defined to issue an error message; `\enablehyperlinks` redefines `\hl@setparam` [606] to be the macro `\@hl@setparam` [237–259].

## 6.1 Default types and options

Default destination type is stored in `\hldest@type`; default link type is stored in `\hl@type`. Default value for an option ⟨*name*⟩ is stored in `\hldest@opt@`⟨*name*⟩ or `\hl@opt@`⟨*name*⟩. These macros are initialized by a hyperlink driver (see Section 3.1, p. 7).

Before `\@hl@setparam` is executed from one of the top-level interface macros, `\hl@param@read@excl` [224–233] checks for the presence of '`!`' following them and remembers its presence by setting the `\if@params@override` switch to `true`. This will later determine whether the current group option list is erased before applying the new option list (see Section 6.2).

If group list is empty (i.e., if no optional argument is given to top-level interface macro), `\@hl@setparam` calls `\hl@setparam@default` [240]. Otherwise `\@hl@setparam` starts scanning the list of groups in the optional parameter and setting group options or types for the groups listed (see Section 6.2). However, if a reserved group with the empty name is encountered in the list of groups, `\hl@setparam@default` is called to set the default options or types [264].

When setting options, `\hl@setparam@default` [292–315] iterates over the option list, calling `\hl@set@opt` [128–143] to set an option (the same macro is used by the explicit hyperlink macros, see Section 4.3, p. 9). When setting type, `\hl@setparam@default` simply defines a macro named `\hl@type` or `\hldest@type` to the requested value.

## 6.2 Group types and options

Group type for a group ⟨*name*⟩ is stored in `\hldest@type@`⟨*name*⟩ [412–420] or `\hl@type@`⟨*name*⟩ [443–455]. Unlike default options, we do not store value for each option for each group in a separate macro. Instead, we store a comma-separated list of ⟨*option*⟩=⟨*value*⟩ assignments for each group ⟨*name*⟩ in the macros `\hldest@opts@`⟨*name*⟩ [421–429] or `\hl@opts@`⟨*name*⟩ [456–466]. These lists contain a subset of the supported options; if assignment for some option is not in the list, the default value (see Section 6.1) for that option will be used.

When the optional parameter is not empty, `\@hl@setparam` starts iterating over the list of groups in the optional parameter using the `\For` loop [248].

Two special group names are reserved and are treated specially: a group named '`*`' is expanded to all defined destgroups (for `\hldesttype` and `\hldestopts`) or linkgroups (for `\hltype` and `\hlopts`); and a group with the empty name, which sets a default type or options.

Upon encountering the '`*`' group, `\hl@do@all@groups` (just a `\gobble` by default) is defined to recursively call `\@hl@setparam` with a list of all defined destgroups or linkgroups for the optional parameter [249]. `\hl@do@all@groups` is called after the `\For` loop is completed [257].

For all other groups `\@hl@setparam` calls `\hl@setparam@group` to parse and set types and options for the group [253]. `\hl@setparam@group` [260–291] first checks for an empty group, handled simply by calling `\hl@setparam@default`. For all other groups, type is set by defining the macro `\hldest@type@`⟨*group_name*⟩ or `\hl@type@`⟨*group_name*⟩ to the requested value; and option list is parsed and set by `\hl@update@opts@with@list`. For option lists, `\hl@setparam@group` also clears the previous value of the group's option list if the switch `\if@params@override` is set to `true` [278] (see p. 12).

`\hl@update@opts@with@list` [316–332] iterates over the user-provided list of option assignments, calling `\hl@update@opts@with@opt` for each. Since one `\For` loop is already in progress at the outer level (within `\@hl@setparam` [248]), the loop in `\hl@update@opts@with@list` is placed within the `\begingroup ... \endgroup` group to avoid clashes with the outer-level `\For`.

`\hl@update@opts@with@opt` [333–380] takes option assignment and constructs a new option list out of the current option list for the group by either updating that option (if the option is already in the list) or adding it at the end of the list (if the option is not yet in the list). This new option list is saved in `\hl@update@new@list` [373] and then assigned in `\hl@update@opts@with@list` as the group's option list [330]. Here again, `\hl@update@opts@with@opt` uses the `\For` loop, which has to be isolated to avoid clashes with an outer-level `\For`.

# 7 Turning hyperlinks on/off

The next group of macros is for turning hyperlinks on and off. It is possible to switch all hyperlinks on the lowest level, so that all hyperlink macros stop creating hyperlinks (see Section 7.1). It is also possible to switch only hyperlinks generated by macros from specific destinations or link groups (see Section 7.2).

`\@hlon`, `\@hloff`, `\@hldeston` and `\@hldestoff` [493–496] are the top-level interface macros for `\@finhlswitch` [506–534] which does the actual parsing of their parameters and switching of hyperlinks. These are what the macros `\hlon`, `\hloff`, `\hldeston` and `\hldestoff` become when the hyperlinks are enabled with the `\enablehyperlinks` macro [619].

## 7.1 Turning low-level hyperlinks on/off

The macros `\@@hlon` [536–539], `\@@hloff` [540–543], `\@@hldeston` [544–546] and `\@@hldestoff` [547–549] affect all hyperlinks. `\@finhlswitch` calls these when the top-level interface macros are called without the optional parameter or with the empty group [508], [521].

Since the macros `\@@hloff` and `\@@hldestoff` are "cheaper" than the full-fledged `\hloff` and `\hldestoff`, they are used extensively by Eplain macros when they need to unconditionally turn off all hyperlinks.

## 7.2 Turning group hyperlinks on/off

Current state of a group ⟨*name*⟩ is stored as an integer value in the macros `\hldest@on@`⟨*name*⟩ [402–411] (for a destgroup) or `\hl@on@`⟨*name*⟩ [431–442] (for a linkgroup). The value of '`0`' (zero) means that the group is turned off; other values mean that the group is turned on.

When the optional parameter (list of groups) is not empty, `\@finhlswitch` starts iterating over the list of groups in the optional parameter using the `\For` loop [516–531]. Two special group names are reserved and are treated specially: a group named '`*`' is expanded to all defined destgroups (for `\hldeston` and `\hldestoff`) or linkgroups (for `\hlon` and `\hloff`); and a group with the empty name, which switches the low-level hyperlink macros (see Section 7.1).

Upon encountering the '`*`' group, `\hl@do@all@groups` (just a `\relax` by default) is defined to recursively call `\@finhlswitch` with a list of all defined destgroups or linkgroups for the optional parameter [517]. `\hl@do@all@groups` is called after the `\For` loop is completed [533].

For all other groups `\@finhlswitch` defines `\hldest@on@`⟨*group_name*⟩ or `\hl@on@`⟨*group_name*⟩ to '`0`' (for `\hldestoff` and `\hloff`) or '`1`' (for `\hldeston` and `\hlon`).

## 8 Acknowledgements

Hypertext support for Eplain described in this document was developed with the help and advice from the Eplain community [27]. Some passages in this document were borrowed from the Eplain manual [7] and the TUGboat article about Eplain [6].

## References

[1] Paul W. Abrahams, Karl Berry, and Kathryn A. Hargreaves. *TEX for the Impatient*. Addison-Wesley, Reading, MA, USA, 1990. Available online from:
http:/tug.org/ftp/tex/impatient
(accessed April 20, 2006).

[2] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook*. Addison-Wesley, Reading, MA, USA, 1985. Available online from:
http://www-cdf.fnal.gov/offline/PostScript/BLUEBOOK.PDF
(accessed April 20, 2006).

[3] Adobe Systems Incorporated. *PDF Reference: Adobe Portable Document Format, Version 1.4*. Addison-Wesley, Reading, MA, USA, third edition, 2001. Available online from http://www.adobe.com.

[4] Adobe Systems Incorporated. ***pdfmark*** *Reference Manual*, October 2, 2005. Available online from http://www.adobe.com.

[5] American Mathematical Society. *User's Guide to* $\mathcal{AMS}$*-TEX*, August 2001.

[6] Karl Berry and Oleg Katsitadze. Eplain 3: Expanded plain TEX. *TUGboat*, 26(3), 2005.

[7] Karl Berry, Oleg Katsitadze, and Steven Smith. *Eplain: Expanded Plain TEX*, December 2005. Latest release version available online from:
http://tug.org/eplain/doc/eplain/index.html
(accessed April 20, 2006). Development version available online from:
http://tug.org/eplain/src/doc/eplain.pdf
(accessed April 20, 2006). Printed version available from:
http://www.lulu.com/content/113810
(accessed April 20, 2006).

[8] Tanmoy Bhattacharya. *The* ***hyperbasics*** *package*. ftp://ftp.tex.ac.uk/tex-archive/support/hypertex/tanmoy/hyperbasics.tex
(accessed April 16, 2006).

[9] Robert J. Chassell and Richard M. Stallman. *GNU Texinfo*, December 29, 2004.

[10] Victor Eijkhout. *The Lollipop macro package*, January 1993. http://www.ctan.org/tex-archive/nonfree/macros/lollipop (accessed April 14, 2006).

[11] *Ghostscript, Ghostview and GSview*, February 24, 2006. http://www.cs.wisc.edu/~ghost (accessed April 20, 2006).

[12] Paul Ginsparg. *The* **lanlmac** *package*, July 1994. http://arxiv.org/hypertex/lanlmac.tex (accessed April 15, 2006).

[13] Shunsaku Hirata and Jin-Hwan Cho. *The DVIPDFMx project*, May 5, 2005. http://project.ktug.or.kr/dvipdfmx (accessed April 15, 2006).

[14] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[15] Donald E. Knuth. *TEX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[16] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1986.

[17] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LATEX2ε*, December 11, 2002.

[18] Kasper Peeters et al. *HyperTEX FAQ*, March 10, 2004. http://arxiv.org/hypertex (accessed April 16, 2006).

[19] Pragma Advanced Document Engineering. *Pragma ADE Web site*, 2006. http://www.pragma-ade.com (accessed April 14, 2006).

[20] Radical Eye Software. *The official dvips home page*. http://www.radicaleye.com/dvips.html (accessed April 15, 2006).

[21] Sebastian Rahtz. *Hypertext Marks in LATEX: The* **hyperref** *Package*, June 1998.

[22] Michael D. Spivak. *The Joy of TEX: A Gourmet Guide to Typesetting with the AMS-TEX Macro Package*. American Mathematical Society, Providence, RI, USA, second edition, 1990.

[23] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen, and Hartmut Henkel. *The pdfTEX User Manual*.

[24] *The DVIPDFM Page*, May 28, 2002. http://gaspra.kettering.edu/dvipdfm (accessed April 17, 2006).

[25] *The pdfTEX testing page*, May 24, 2004. http://pdftex.sarovar.org (accessed April 20, 2006).

[26] TUG. *The Eplain home page*, 2006. http://tug.org/eplain (accessed April 16, 2006).

[27] TUG. *The Eplain mailing list*, 2006. http://tug.org/mailman/listinfo/tex-eplain (accessed April 15, 2006).

## Appendix. Source listing

Listing 8.1 contains the hyperlink part of the Eplain's `xeplain.tex` source file.

Listing 8.1

Source code of the hyperlink support for Eplain

```
%
% Hypertext links support.
%
% Hyperlink destinations (driver-independent code).
%
% \hldest{TYPE}{OPTIONS}{LABEL} defines a hyperlink destination
% LABEL.  OPTIONS is a comma-separated list of option assignments of
% the form 'opt=value'; permitted values for TYPE and OPTIONS depend
% on selected hyperlink driver.
%
% \hldest will be \let to \@hldest by \enablehyperlinks.  TYPE,
% OPTIONS and LABEL will be read by \hl@getparam.
\def\@hldest{%
  \def\hl@prefix{hldest}%
  \let\after@hl@getparam\hldest@aftergetparam
  % Start the group which will isolate option settings.  It will be
  % ended in \hldest@aftergetparam
  \begingroup
    \hl@getparam
}%
% This actually produces hyperlink destination.  It will be called at
% the end of \hl@getparam, after the parameters are parsed.
\def\hldest@aftergetparam{%
  \ifvmode
    % In vertical mode we don't raise the destination, so it can go
    % directly into the vertical list.
    \hldest@driver
  \else
    % In horizontal mode, the destination is raised \hldest@opt@raise
    % above the baseline and placed inside a zero-width/height/depth
    % box; the box is surrounded by \allowhyphens in case it is
    % placed next to a word, to allow hyphenation of that word.
    \allowhyphens
    \smash{\ifx\hldest@opt@raise\empty \else \raise\hldest@opt@raise\fi
             \hbox{\hldest@driver}}%
    \allowhyphens
  \fi
  % End the group which was isolating option settings (it was started
  % in \@hldest).
  \endgroup
}%
%
% Hyperlinks (driver-independent code).
%
% \hlstart{TYPE}{OPTIONS}{LABEL} starts a hyperlink to destination
% LABEL.  OPTIONS is a comma-separated list of option assignments of
% the form 'opt=value'; permitted values of TYPE and OPTIONS depend on
% selected hyperlink driver.  End the link with \hlend.
%
% \hlstart will be \let to \@hlstart by \enablehyperlinks.  TYPE,
% OPTIONS and LABEL will be read by \hl@getparam.
\def\@hlstart{%
```

```
    \leavevmode
    \def\hl@prefix{hl}%
    \let\after@hl@getparam\hlstart@aftergetparam
    % Start the group which will isolate option settings and color
    % changes.  It will be ended in \@hlend
    \begingroup
      \hl@getparam
  }%
  %
  \def\hlstart@aftergetparam{%
    % Set the color for the link.
    \ifx\color\undefined \else
      \ifx\hl@opt@color\empty \else
        \ifx\hl@opt@colormodel\empty
          \edef\temp{\noexpand\color{\hl@opt@color}}%
        \else
          \edef\temp{\noexpand\color[\hl@opt@colormodel]{\hl@opt@color}}%
        \fi
        \temp
      \fi
    \fi
    % Call the driver.
    \hl@driver
  }%
  % \hlend will be \let to \@hlend by \enablehyperlinks.  \@hlend will
  % be defined by a driver.
  %
  % Macros which are used commonly by \hldest and \hlstart to parse and
  % save parameters.  \hl@prefix must be set to `hldest' by \hldest and
  % to `hl' by \hlstart.
  %
  % \hl@getparam{TYPE}{OPTIONS}{LABEL} reads, parses and saves the
  % parameters for \hldest or \hlstart into \@hltype, \hl[dest]@opt@...
  % and \@hllabel.  After doing that it calls \after@hl@getparam which
  % should be defined by \@hldest and \@hlstart to produce destination /
  % link using the saved parameters.
  \def\hl@getparam#1#2{% We'll read #3 (LABEL) later.
    % Save TYPE in \@hltype (use default if empty).
    \edef\@hltype{#1}%
    \ifx\@hltype\empty
      \expandafter\let\expandafter\@hltype
        \csname \hl@prefix @type\endcsname
    \fi
    % For each supported destination / link type TYPE, a driver should
    % define \hl[dest]@typeh@TYPE handler.
    \expandafter\ifx\csname \hl@prefix @typeh@\@hltype\endcsname \relax
      \errmessage{Invalid hyperlink type `\@hltype'}%
    \fi
    % \for will expand #2 once so user can pass a shortcut macro.  We
    % also ignore empty \hl@arg, so that the following
    %     \hldest{TYPE}{\myopt,height=200pt}{LABEL}
    % would be legal even when \myopt happens to be empty.
    \For\hl@arg:=#2\do{%
      \ifx\hl@arg\empty \else
        \expandafter\hl@set@opt\hl@arg=,%
      \fi
    }%
    % Now read the third argument, LABEL.  Do so inside a group with
```

```
[111]   % \uncatcodespecials, to allow `#' and `~' in it (LABEL can be a URL
[112]   % for some link types).
[113]   \bgroup
[114]     \uncatcodespecials
[115]     \catcode`\{=1 \catcode`\}=2
[116]     \@hl@getparam
[117] }%
[118] %
[119] \def\@hl@getparam#1{%
[120]   \egroup
[121]   % Save LABEL in \@hllabel.
[122]   \edef\@hllabel{#1}%
[123]   % Execute the commands specific to destination / link
[124]   \after@hl@getparam
[125]   % Ignore spaces after \hlstart and \hldest.
[126]   \ignorespaces
[127] }%
[128] % Parse and set a (default, not a group) option.
[129] \def\hl@set@opt#1=#2,{%
[130]   % For each supported option OPTION, a driver should define
[131]   % \hl[dest]@opt@OPTION.
[132]   \expandafter\ifx\csname \hl@prefix @opt@#1\endcsname \relax
[133]     \errmessage{Invalid hyperlink option `#1'}%
[134]   \fi
[135]   % Save the value of the option.
[136]   \if ,#2, % if #2 is empty, complain.
[137]     \errmessage{Missing value for option `#1'}%
[138]   \else
[139]     % Remove a trailing =.
[140]     \def\temp##1={##1}%
[141]     \expandafter\edef\csname \hl@prefix @opt@#1\endcsname{\temp#2}%
[142]   \fi
[143] }%
[144] % \hl{dest,start,end}@impl{GROUP}{LABEL} will generate `implicit'
[145] % destination / hyperlink, if the user has not turned off this kind of
[146] % implicit destinations / hyperlinks.  This is used by Eplain's
[147] % cross-reference macros.
[148] \def\hldest@impl#1{%
[149]   \expandafter\ifcase\csname hldest@on@#1\endcsname
[150]     \relax\expandafter\gobble
[151]   \else
[152]     \toks@=\expandafter{\csname hldest@type@#1\endcsname}%
[153]     \toks@ii=\expandafter{\csname hldest@opts@#1\endcsname}%
[154]     \edef\temp{\noexpand\hldest{\the\toks@}{\the\toks@ii}}%
[155]     \expandafter\temp
[156]   \fi
[157] }%
[158] \def\hlstart@impl#1{%
[159]   \expandafter\ifcase\csname hl@on@#1\endcsname
[160]     % Still produce \leavevmode, to be consistent with \hloff.
[161]     \leavevmode\expandafter\gobble
[162]   \else
[163]     \toks@=\expandafter{\csname hl@type@#1\endcsname}%
[164]     \toks@ii=\expandafter{\csname hl@opts@#1\endcsname}%
[165]     \edef\temp{\noexpand\hlstart{\the\toks@}{\the\toks@ii}}%
[166]     \expandafter\temp
[167]   \fi
[168] }%
```

```
[169]  \def\hlend@impl#1{%
[170]    \expandafter\ifcase\csname hl@on@#1\endcsname
[171]    \else
[172]      \hlend
[173]    \fi
[174]  }%
```

```
%
% Setting options and types.
%
\def\hl@asterisk@word{*}%
\def\hl@opts@word{opts}%
\newif\if@params@override
% We define hyperlink / destination groups.  A group is a macro or a
% group of macros which implicitly generate hyperlink / destination.
% The user can set parameters for each group individually, as well as
% the default parameters, with the macros defined below.  Group
% settings will override the default hyperlink / destination
% parameters.
\def\hldest@groups{definexref,xrdef,li,eq,bib,foot,footback,idx}%
\def\hl@groups{ref,xref,eq,cite,foot,footback,idx,url,hrefint,hrefext}%
% \hldesttype [GROUPS]{VALUE}
% \hldestopts [GROUPS]{VALUE}
% \hltype     [GROUPS]{VALUE}
% \hlopts     [GROUPS]{VALUE}
%
% Set hyperlink or destination parameter (type / opts) to VALUE for
% each group in GROUPS.  An empty 'group' will set default value for
% the parameter.  A star (*) 'group' stands for all groups (except the
% empty 'group').  If the macro is followed by an exclamation mark
% (like \hlopts!...), the parameters will be overridden; otherwise,
% they will be updated (this has effect only on group option list).
\def\hldesttype{%
  \def\hl@prefix{hldest}%
  \def\hl@param{type}%
  \let\hl@all@groups\hldest@groups
  \futurelet\hl@excl\hl@param@read@excl
}%
\def\hldestopts{%
  \def\hl@prefix{hldest}%
  \def\hl@param{opts}%
  \let\hl@all@groups\hldest@groups
  \futurelet\hl@excl\hl@param@read@excl
}%
\def\hltype{%
  \def\hl@prefix{hl}%
  \def\hl@param{type}%
  \let\hl@all@groups\hl@groups
  \futurelet\hl@excl\hl@param@read@excl
}%
\def\hlopts{%
  \def\hl@prefix{hl}%
  \def\hl@param{opts}%
  \let\hl@all@groups\hl@groups
  \futurelet\hl@excl\hl@param@read@excl
}%
\def\hl@param@read@excl{%
  \ifx!\hl@excl
    \let\next\hl@param@read@opt@arg
    \@params@overridetrue
  \else
    \def\next{\hl@param@read@opt@arg{!}}%
    \@params@overridefalse
  \fi
  \next
```

```
}%
\def\hl@param@read@opt@arg#1{% #1 is the '!', ignore it.
  \@getoptionalarg\hl@setparam
}%
% Set the parameter \hl@param to #1 for each group in \@optionalarg.
% This will become \hl@setparam in \enablehyperlinks.
\def\@hl@setparam#1{%
  \ifx\@optionalarg\empty
    \hl@setparam@default{#1}% Set default.
  \else
    % If we find an asterisk in the list, we have no choice but to
    % finish the list and then call \hl@setparam again, now with
    % \hl@all@groups for the list of groups.
    \let\hl@do@all@groups\gobble
%
    \For\hl@group:=\@optionalarg\do{%
      \ifx\hl@group\hl@asterisk@word
        \def\hl@do@all@groups{\let\@optionalarg\hl@all@groups
                              \hl@setparam}%
      \else
        \hl@setparam@group{#1}%
      \fi
    }%
%
    \hl@do@all@groups{#1}%
  \fi
}%
% Set a parameter (\hl@param) for one group (\hl@group) to the value
% (#1).  The group may be empty, in which case we call
% \hl@setparam@default
\def\hl@setparam@group#1{%
  \ifx\hl@group\empty
    \hl@setparam@default{#1}%
  \else
    \expandafter\ifx\csname\hl@prefix @\hl@param @\hl@group\endcsname
                    \relax
      \errmessage{Hyperlink group `\hl@prefix:\hl@param:\hl@group'
                  is not defined}%
    \fi
    \ifx\hl@param\hl@opts@word
      % For the `opts' parameter, we want to expand the first token of
      % #1 once, in case the user passed a macro containing the option
      % list.  Even if we simply need to override the old option list,
      % we still call \hl@update@opts@with@list to go through the
      % options and trim possible leading space token in option keys.
      \if@params@override
        \expandafter\let\csname\hl@prefix @\hl@param @\hl@group\endcsname
                        \empty
      \fi
      \hl@update@opts@with@list{#1}% #1 will be used in the \for
                                   % loop, so it will be expanded once.
    \else
      % Do not use \edef here to define the parameter, so the user can
      % define it to, e.g., \normalbaselineskip, and make the parameter
      % adjustable to a situation.
      \ece\def{\hl@prefix @\hl@param @\hl@group}{#1}%
    \fi
  \fi
```

```
}%
% Set default parameter values.  We have to treat 'opts' (list of
% options) specially, because for option defaults we don't store a
% list of options (like we do for the group options) but set each
% option individually.
\def\hl@setparam@default#1{%
  \ifx\hl@param\hl@opts@word
    % 'opts'.
    \For\hl@opt:=#1\do{%
      \ifx\hl@opt\empty \else
        \expandafter\hl@set@opt\hl@opt=,%
      \fi
    }%
  \else
    % Everything except 'opts'.
    \expandafter\ifx\csname\hl@prefix @\hl@param\endcsname\relax
      \message{Default hyperlink parameter '\hl@prefix:\hl@param'
               is not defined}%
    \fi
    % Should not use \edef, so the user could define this to, e.g.,
    % \normalbaselineskip, to make the parameter adjustable to a
    % situation.
    \ece\def{\hl@prefix @\hl@param}{#1}%
  \fi
}%
% For each option in the list (#1), call \hl@update@opts@with@opt to
% update the group's option list (\csname\hl@prefix @opts@
% \hl@group\endcsname) with this new option.
\def\hl@update@opts@with@list#1{%
  % Start with the current list of the group.
  \global\expandafter\let\expandafter\hl@update@new@list
    \csname \hl@prefix @opts@\hl@group\endcsname
  % We have to isolate the \for loop inside a (TeX) group, to avoid
  % clashes with the loop in \hl@setparam
  \begingroup
    \For\hl@opt:=#1\do{%
      \hl@update@opts@with@opt
    }%
  \endgroup
  % Save the final list back in the option list for the group.
  \ece\let{\hl@prefix @opts@\hl@group}\hl@update@new@list
}%
% Go through the option list (\hl@update@new@list) and construct the
% new list (in \hl@update@new@list), replacing the old definition of
% the option with the new one (\hl@opt).
\def\hl@update@opts@with@opt{%
  % Save the old list and the new option.
  \global\let\hl@update@old@list\hl@update@new@list
  \global\let\hl@update@new@list\empty
  \global\let\hl@update@new@opt\hl@opt
  % Get the key of the new option and save it.
  \expandafter\hl@parse@opt@key\hl@opt=,%
  \let\hl@update@new@key\hl@update@key
  % We will set this to real comma after the first entry.
  \global\let\hl@update@comma\empty
  % We have to isolate the \for loop inside a (TeX) group, to avoid
  % clashes with the loop in \hl@update@opts@with@list
  \begingroup
```

```
[349]      \for\hl@opt:=\hl@update@old@list\do{%
[350]        \ifx\hl@opt\empty \else % Skip empty 'options'.
[351]          % Get the key of this option.
[352]          \expandafter\hl@parse@opt@key\hl@opt=,%
[353]          % If the key matches, replace the option definition with the
[354]          % new definition, otherwise, repeat the old definition.
[355]          \toks@=\expandafter{\hl@update@new@list}%
[356]          \ifx\hl@update@key\hl@update@new@key
[357]            \ifx\hl@update@new@opt\empty \else % Skip multiple options.
[358]              \toks@ii=\expandafter{\hl@update@new@opt}%
[359]              \xdef\hl@update@new@list{\the\toks@\hl@update@comma
[360]                                       \the\toks@ii}%
[361]              \global\let\hl@update@new@opt\empty
[362]              \global\def\hl@update@comma{,}%
[363]            \fi
[364]          \else
[365]            \toks@ii=\expandafter{\hl@opt}%
[366]            \xdef\hl@update@new@list{\the\toks@\hl@update@comma
[367]                                     \the\toks@ii}%
[368]            \global\def\hl@update@comma{,}%
[369]          \fi
[370]        \fi
[371]      }%
[372]    \endgroup
[373]    % If nothing was replaced, add the new option to the end of the new
[374]    % list.
[375]    \ifx\hl@update@new@opt\empty \else
[376]      \toks@=\expandafter{\hl@update@new@list}%
[377]      \toks@ii=\expandafter{\hl@update@new@opt}%
[378]      \xdef\hl@update@new@list{\the\toks@\hl@update@comma\the\toks@ii}%
[379]    \fi
[380]  }%
[381]  % Parse the key of the option and save it in \hl@update@key
[382]  \def\hl@parse@opt@key#1=#2,{\def\hl@update@key{#1}}%
```

```
%
% Default and group parameters (options and types).
%
% Option 'raise' will determine how much to raise hyperlink
% destinations above the baseline.  It will be supported by all
% drivers, since it is handled outside the drivers, in
% \hldest@aftergetparam.
\def\hldest@opt@raise{\normalbaselineskip}%
% Options 'colormodel' and 'color' will also be handled outside the
% drivers, in \hlstart@aftergetparam.
\def\hl@opt@colormodel{cmyk}%
\def\hl@opt@color{0.28,1,1,0.35}%
%
% Parameters for destinations and links produced implicitly by
% cross-reference macros.  Note that each driver will additionally
% define \hldest@type and \hl@type parameters which will be used when
% one of the below is empty, and default values for destination and
% link options (which are driver-specific).
%
% Destination on/off flags (0=off, 1=on).  Changing them here has no
% effect, modify \enablehyperlinks to set defaults.
\def\hldest@on@definexref{0}%
\def\hldest@on@xrdef{0}%
\def\hldest@on@li{0}%
\def\hldest@on@eq{0}% \eqdef and friends
\def\hldest@on@bib{0}% \biblabelprint (BibTeX)
\def\hldest@on@foot{0}% \footnote / \numberedfootnote
\def\hldest@on@footback{0}% back-ref for \footnote / \numberedfootnote
\def\hldest@on@idx{0}% both 'page' dests and 'exact' dests
% Types of destinations.
\let\hldest@type@definexref\empty
\let\hldest@type@xrdef\empty
\let\hldest@type@li\empty
\let\hldest@type@eq\empty % \eqdef and friends
\let\hldest@type@bib\empty % \biblabelprint (BibTeX)
\let\hldest@type@foot\empty % \footnote / \numberedfootnote
\let\hldest@type@footback\empty % back-ref for \footnote/\numberedfootnote
\let\hldest@type@idx\empty % both 'page' dests and 'exact' dests
% Options for destinations.
\let\hldest@opts@definexref\empty
\let\hldest@opts@xrdef\empty
\let\hldest@opts@li\empty
\def\hldest@opts@eq{raise=1.7\normalbaselineskip}% \eqdef and friends
\let\hldest@opts@bib\empty % \biblabelprint (BibTeX)
\let\hldest@opts@foot\empty % \footnote / \numberedfootnote
\let\hldest@opts@footback\empty % back-ref for \footnote/\numberedfootnote
\let\hldest@opts@idx\empty % both 'page' dests and 'exact' dests
%
% Hyperlink on/off flags (0=off, 1=on).  Changing them here has no
% effect, modify \enablehyperlinks to set defaults.
\def\hl@on@ref{0}% \refn and \xrefn, \ref, \refs
\def\hl@on@xref{0}%
\def\hl@on@eq{0}% \eqref and \eqrefn
\def\hl@on@cite{0}% \cite (BibTeX)
\def\hl@on@foot{0}% \footnote / \numberedfootnote
\def\hl@on@footback{0}% back-ref for \footnote/\numberedfootnote
\def\hl@on@idx{0}%
\def\hl@on@url{0}% \url from url.sty
```

```
\def\hl@on@hrefint{0}% \href with internal #labels
\def\hl@on@hrefext{0}% \href with external labels (URLs)
% Types of links.
\let\hl@type@ref\empty % \refn and \xrefn, \ref, \refs
\let\hl@type@xref\empty
\let\hl@type@eq\empty % \eqref and \eqrefn
\let\hl@type@cite\empty % \cite (BibTeX)
\let\hl@type@foot\empty % \footnote / \numberedfootnote
\let\hl@type@footback\empty % back-ref for \footnote/\numberedfootnote
\let\hl@type@idx\empty
\let\hl@type@url\empty % \url from url.sty (this will be set to `url' by
                       % drivers which support the `url' type)
\let\hl@type@hrefint\empty % \href with internal #labels
\let\hl@type@hrefext\empty % \href with external labels (URLs) (this
      % will be set to `url' by drivers which support the `url' type)
% Options for links.
\let\hl@opts@ref\empty % \refn and \xrefn, \ref, \refs
\let\hl@opts@xref\empty
\let\hl@opts@eq\empty % \eqref and \eqrefn
\let\hl@opts@cite\empty % \cite (BibTeX)
\let\hl@opts@foot\empty % \footnote / \numberedfootnote
\let\hl@opts@footback\empty % back-ref for \footnote/\numberedfootnote
\let\hl@opts@idx\empty
\let\hl@opts@url\empty % \url from url.sty
\let\hl@opts@hrefint\empty % \href with internal #labels
\let\hl@opts@hrefext\empty % \href with external labels (URLs)
```

```
%
% \@hlon[GROUPS]
% \@hloff[GROUPS]
% \@hldeston[GROUPS]
% \@hldestoff[GROUPS]
% \@@hlon
% \@@hloff
% \@@hldeston
% \@@hldestoff
%
% Macros to switch hyperlinks / destinations on/off.
%
% The optional arg is the list of groups.  It can contain a star (*)
% which will make the macros affect all groups (but not the low-level
% macros \hlstart, \hlend and \hldest).
%
% \@hlon, \@hldeston, \@hloff and \@hldestoff will turn low-level
% macros on/off only when they are used either without the optional
% arg or with an empty 'group' in the optional arg, otherwise only the
% specified groups are affected.
%
% The single-'@' variants (\@hl...) are for the user.  In your macros,
% if you want to (temporarily) turn low-level macros on/off, it's
% better to use the double-'@' variants (\@@hl...), because they are
% much faster and won't clobber \@optionalarg or anything else.
%
\def\@hlon{\@hlonoff@value@stub{hl}\@@hlon1 }%
\def\@hloff{\@hlonoff@value@stub{hl}\@@hloff0 }%
\def\@hldeston{\@hlonoff@value@stub{hldest}\@@hldeston1 }%
\def\@hldestoff{\@hlonoff@value@stub{hldest}\@@hldestoff0 }%
%
\def\@hlonoff@value@stub#1#2#3{%
  \def\hl@prefix{#1}%
  \let\hl@on@empty#2%
  \def\hl@value{#3}%
  \expandafter\let\expandafter\hl@all@groups
    \csname \hl@prefix @groups\endcsname
  \@getoptionalarg\@finhlswitch
}%
%
\def\@finhlswitch{%
  \ifx\@optionalarg\empty
    \hl@on@empty
  \fi
  % If we find an asterisk in the list, we have no choice but to
  % finish the list and then call \@finhlswitch again, now with
  % \hl@all@groups for the list of groups.
  \let\hl@do@all@groups\relax
%
  \For\hl@group:=\@optionalarg\do{%
    \ifx\hl@group\hl@asterisk@word
      \let\@optionalarg\hl@all@groups
      \let\hl@do@all@groups\@finhlswitch
    \else
      \ifx\hl@group\empty
        \hl@on@empty
      \else
        \expandafter\ifx\csname\hl@prefix @on@\hl@group\endcsname \relax
```

```
[525]            \errmessage{Hyperlink group `\hl@prefix:on:\hl@group'
[526]                        is not defined}%
[527]          \fi
[528]          \ece\edef{\hl@prefix @on@\hl@group}{\hl@value}%
[529]        \fi
[530]      \fi
[531]    }%
[532]  %
[533]    \hl@do@all@groups
[534]  }%
[535]  % Turn low-level macros on/off.
[536]  \def\@@hlon{%
[537]    \let\hlstart\@hlstart
[538]    \let\hlend\@hlend
[539]  }%
[540]  \def\@@hloff{%
[541]    \def\hlstart##1##2##3{\leavevmode\ignorespaces}%
[542]    \let\hlend\relax
[543]  }%
[544]  \def\@@hldeston{%
[545]    \let\hldest\@hldest
[546]  }%
[547]  \def\@@hldestoff{%
[548]    \def\hldest##1##2##3{\ignorespaces}%
[549]  }%
```

```
[550]  %
[551]  % Hyperlink drivers.
[552]  %
[553]  % \enablehyperlinks[OPTIONS] will enable hyperlinks.  OPTIONS is a
[554]  % list of comma-separated options.  An option is one of the following:
[555]  %
[556]  %     idxexact      Point index links to exact locations of the term
[557]  %     idxpage       Point index links to pages with the term (default)
[558]  %     idxnone       No links for index entries
[559]  %     <driver-name> Force the hyperlink driver
[560]  %
[561]  % If <driver-name> is omitted, appropriate driver will be detected, if
[562]  % possible; if not, we fall back on `hypertex'.
[563]  \def\hl@idxexact@word{idxexact}%
[564]  \def\hl@idxpage@word{idxpage}%
[565]  \def\hl@idxnone@word{idxnone}%
[566]  \def\hl@raw@word{raw}%
[567]  %
[568]  \def\enablehyperlinks{\@getoptionalarg\@finenablehyperlinks}%
[569]  \def\@finenablehyperlinks{%
[570]    \let\hl@selecteddriver\empty
[571]    % By default we generate `idxpage' index hyperlinks.
[572]    \def\hldest@place@idx{0}%
[573]    % Go through the option list.
[574]    \for\hl@arg:=\@optionalarg\do{%
[575]      \ifx\hl@arg\hl@idxexact@word
[576]        \def\hldest@place@idx{1}%
[577]      \else
[578]        \ifx\hl@arg\hl@idxnone@word
[579]          \def\hldest@place@idx{-1}%
[580]        \else
[581]          \ifx\hl@arg\hl@idxpage@word
[582]            \def\hldest@place@idx{0}%
[583]          \else
[584]            \let\hl@selecteddriver\hl@arg
[585]          \fi
[586]        \fi
[587]      \fi
[588]    }%
[589]    % Check the driver name.
[590]    \ifx\hl@selecteddriver\empty
[591]      % The user did not specify a driver, detect.
[592]      \ifpdf
[593]        \def\hl@selecteddriver{pdftex}%
[594]        \message{^^JEplain: using `pdftex' hyperlink driver.}%
[595]      \else
[596]        \def\hl@selecteddriver{hypertex}%
[597]        \message{^^JEplain: using `hypertex' hyperlink driver.}%
[598]      \fi
[599]    \else
[600]      % Check that the requested driver's initialization routine is
[601]      % available.
[602]      \expandafter\ifx\csname hldriver@\hl@selecteddriver\endcsname \relax
[603]        \errmessage{No hyperlink driver `\hl@selecteddriver' available}%
[604]      \fi
[605]    \fi
[606]    % Enable \hltype, \hlopts, \hldest and \hldestopts now (the driver's
[607]    % initialization routine may change this).
```

```
[608]     \let\hl@setparam\@hl@setparam
[609]     % Call the driver's initialization routine.
[610]     \csname hldriver@\hl@selecteddriver\endcsname
[611]     % Driver should not be changed later.
[612]     \def\@finenablehyperlinks{\errmessage{Hyperlink driver
[613]                                  `\hl@selecteddriver' already selected}}%
[614]     % Free memory taken up by the drivers.
[615]     \let\hldriver@nolinks\undefined
[616]     \let\hldriver@hypertex\undefined
[617]     \let\hldriver@pdftex \undefined
[618]     \let\hldriver@dvipdfm\undefined
[619]     % The user can use these to turn the links / destinations on/off
[620]     % (see comments to the driver `nolinks').
[621]     \let\hloff\@hloff
[622]     \let\hlon\@hlon
[623]     \let\hldestoff\@hldestoff
[624]     \let\hldeston\@hldeston
[625]     % By default turn everything on except the footnotes.
[626]     \hlon[*,]\hloff[foot,footback]%
[627]     \hldeston[*,]\hldestoff[foot,footback]%
[628]   }%
[629]   %
[630]   % Driver `nolinks'.
[631]   %
[632]   % Select this driver to suppress any hyperlinks / destinations in your
[633]   % document.
[634]   %
[635]   % NOTE:  selecting this driver is quite different from not selecting
[636]   % any driver at all, or from selecting some driver and then turning
[637]   % off links and destinations for the entire document with \hloff and
[638]   % \hldestoff.
[639]   %
[640]   % The purpose of \hldestoff and \hloff is to mark (parts) of document
[641]   % where links should never appear.  (Imagine you want to prevent a
[642]   % cross-referencing macro from generating a link at a certain spot in
[643]   % your document.)
[644]   %
[645]   % If instead you have prepared a document with links and just want to
[646]   % compile a version without the links, it is better to select the
[647]   % driver `nolinks'.  This will ensure that spacing and pagebreaking
[648]   % will be the same as what you were getting with hyperlinks enabled.
[649]   %
[650]   % The reason for this is that hyperlinks are produced by \special
[651]   % commands.  Each \special is placed inside a whatsit which may
[652]   % introduce a legitimate breakpoint at places where none would exist
[653]   % without the whatsit.  The macros \hldestoff and \hloff disable
[654]   % hyperlink macros so drastically that no whatsits are produced.
[655]   %
[656]   % On the other hand, `nolinks' driver does not completely disable
[657]   % hyperlink macros.  Instead, it defines them to write to the log
[658]   % file (what gets written is not really important).  This will produce
[659]   % the whatsits imitating the whatsits from the \special's.  (This
[660]   % trick was borrowed from graphics bundle.)
[661]   %
[662]   % Another reason for using `nolinks' is that in horizontal mode
[663]   % \hldest places destinations inside zero-width/height/depth boxes.
[664]   % When you say \hldestoff, \hldest will omit both destination specs
[665]   % and these boxes.  The missing boxes can cause typesetting to be
```

```
% inconsistent with what you were getting with destinations enabled.
% Again, 'nolinks' driver helps here by defining \hldest to still
% produce the empty boxes.
%
% Additionally, 'nolinks' driver defines the \hldesttype, \hldestopts,
% \hltype, \hlopts macros to gobble their parameters, to avoid error
% messages about "unknown" options and types under the 'nolinks'
% driver.
\def\hldriver@nolinks{%
  \def\@hldest##1##2##3{%
    \edef\temp{\write-1{hldest: ##3}}%
    \ifvmode
      \temp
    \else
      \allowhyphens
      \expandafter\smash\expandafter{\temp}%
      \allowhyphens
    \fi
    \ignorespaces
  }%
  \def\@hlstart##1##2##3{%
    \leavevmode
    \begingroup % Start the color group.
    \edef\temp{\write-1{hlstart: ##3}}%
    \temp
    \ignorespaces
  }%
  \def\@hlend{%
    \edef\temp{\write-1{hlend}}%
    \temp
    \endgroup % End the color group from \@hlstart.
  }%
  % Make \hltype, \hlopts, \hldesttype and \hldestopts ignore their
  % parameters.
  \let\hl@setparam\gobble
}%
%
% Driver 'hypertex'.
%
\expandafter\def\expandafter\hlhash\expandafter{\string#}%
%
\def\hldriver@hypertex{%
  %
  % Hyperlink destinations.
  %
  % Default type.
  \def\hldest@type{xyz}%
  % Set defaults for the options (this also tells \hl@set@opt what
  % options we support).  (We do not define \hldest@opt@raise,
  % \hl@opt@colormodel and \hl@opt@color, they are defined and used
  % outside the drivers.)
  \let\hldest@opt@cmd \empty
  % Multiplexer for all supported destination types.
  \def\hldest@driver{%
    % Special case for 'raw' destinations.
    \ifx\@hltype\hl@raw@word
      \csname \hldest@opt@cmd \endcsname
    \else
```

```
[724]       \special{html:<a name="\@hllabel">}\special{html:</a>}%
[725]     \fi
[726]   }%
[727]   % Define handlers for each supported destination type (this also
[728]   % tells \hl@getparam what types we support).
[729]   \let\hldest@typeh@raw \empty
[730]   \let\hldest@typeh@xyz \empty
[731]   %
[732]   % Hyperlinks.
[733]   %
[734]   % Default type.
[735]   \def\hl@type{name}%
[736]   % We support 'url' hyperlinks, so set some group types.
[737]   \ifx\hl@type@url\empty
[738]     \def\hl@type@url{url}%
[739]   \fi
[740]   \ifx\hl@type@hrefext\empty
[741]     \def\hl@type@hrefext{url}%
[742]   \fi
[743]   % Set defaults for the options (this also tells \hl@set@opt what
[744]   % options we support).
[745]   \let\hl@opt@cmd  \empty
[746]   \let\hl@opt@ext  \empty
[747]   \let\hl@opt@file \empty
[748]   % Multiplexer for all supported link types.
[749]   \def\hl@driver{%
[750]     % Special case for 'raw' links.
[751]     \ifx\@hltype\hl@raw@word
[752]       \csname \hl@opt@cmd \endcsname
[753]     \else
[754]       % Construct common preamble of a link.
[755]       \def\hlstart@preamble{html:<a href="}%
[756]       % Call the handler.
[757]       \csname hl@typeh@\@hltype\endcsname
[758]     \fi
[759]   }%
[760]   % Define handlers for each supported link type (this also tells
[761]   % \hl@getparam what types we support).
[762]   \let\hl@typeh@raw \empty
[763]   \def\hl@typeh@name{\special{\hlstart@preamble \hlhash\@hllabel">}}%
[764]   \def\hl@typeh@filename{%
[765]     \special{%
[766]       \hlstart@preamble
[767]         file:\hl@opt@file\hl@opt@ext
[768]         \ifempty\@hllabel \else \hlhash\@hllabel\fi
[769]       ">%
[770]     }%
[771]   }%
[772]   \def\hl@typeh@url{%
[773]     \special{%
[774]       \hlstart@preamble
[775]         \@hllabel
[776]       ">%
[777]     }%
[778]   }%
[779]   %
[780]   \def\@hlend{\special{html:</a>}\endgroup}% The group from \@hlstart.
[781] }%
```

```
[782]  %
[783]  % Driver 'pdftex'.
[784]  %
[785]  \def\hldriver@pdftex{%
[786]  \ifpdf % PDF output is enabled.
[787]    %
[788]    % Hyperlink destinations.
[789]    %
[790]    % Default type.
[791]    \def\hldest@type{xyz}%
[792]    % Set defaults for the options (this also tells \hl@set@opt what
[793]    % options we support).  (We do not define \hldest@opt@raise,
[794]    % \hl@opt@colormodel and \hl@opt@color, they are defined and used
[795]    % outside the drivers.)
[796]    \let\hldest@opt@width  \empty
[797]    \let\hldest@opt@height \empty
[798]    \let\hldest@opt@depth  \empty
[799]    \let\hldest@opt@zoom   \empty
[800]    \let\hldest@opt@cmd    \empty
[801]    % Multiplexer for all supported destination types.
[802]    \def\hldest@driver{%
[803]      % Special case for 'raw' destinations.
[804]      \ifx\@hltype\hl@raw@word
[805]        \csname \hldest@opt@cmd \endcsname
[806]      \else
[807]        \pdfdest name{\@hllabel}\@hltype
[808]          \csname hldest@typeh@\@hltype\endcsname
[809]      \fi
[810]    }%
[811]    % Define handlers for each supported destination type (this also
[812]    % tells \hl@getparam what types we support).
[813]    \let\hldest@typeh@raw   \empty
[814]    \let\hldest@typeh@fit   \empty
[815]    \let\hldest@typeh@fith  \empty
[816]    \let\hldest@typeh@fitv  \empty
[817]    \let\hldest@typeh@fitb  \empty
[818]    \let\hldest@typeh@fitbh \empty
[819]    \let\hldest@typeh@fitbv \empty
[820]    \def\hldest@typeh@fitr{%
[821]      \ifx\hldest@opt@width  \empty \else width  \hldest@opt@width  \fi
[822]      \ifx\hldest@opt@height \empty \else height \hldest@opt@height \fi
[823]      \ifx\hldest@opt@depth  \empty \else depth  \hldest@opt@depth  \fi
[824]    }%
[825]    \def\hldest@typeh@xyz{%
[826]      \ifx\hldest@opt@zoom\empty \else zoom \hldest@opt@zoom \fi
[827]    }%
[828]    %
[829]    % Hyperlinks.
[830]    %
[831]    % Default type.
[832]    \def\hl@type{name}%
[833]    % We support 'url' hyperlinks, so set some group types.
[834]    \ifx\hl@type@url\empty
[835]      \def\hl@type@url{url}%
[836]    \fi
[837]    \ifx\hl@type@hrefext\empty
[838]      \def\hl@type@hrefext{url}%
[839]    \fi
```

```
[840]    % Set defaults for the options (this also tells \hl@set@opt what
[841]    % options we support).
[842]    \let\hl@opt@width   \empty
[843]    \let\hl@opt@height  \empty
[844]    \let\hl@opt@depth   \empty
[845]    \def\hl@opt@bstyle  {S}%
[846]    \def\hl@opt@bwidth  {1}%
[847]    \let\hl@opt@bcolor  \empty
[848]    \let\hl@opt@hlight  \empty
[849]    \let\hl@opt@bdash   \empty
[850]    \let\hl@opt@pagefit \empty
[851]    \let\hl@opt@cmd     \empty
[852]    \let\hl@opt@file    \empty
[853]    \let\hl@opt@newwin  \empty
[854]    % Multiplexer for all supported link types.
[855]    \def\hl@driver{%
[856]      % Special case for 'raw' links.
[857]      \ifx\@hltype\hl@raw@word
[858]        \csname \hl@opt@cmd \endcsname
[859]      \else
[860]        % See if we will construct a /BS spec.  We want to bother only
[861]        % if any of \hl@opt@bstyle, \hl@opt@bwidth and \hl@opt@bdash is
[862]        % not empty.
[863]        \let\hl@BSspec\relax % construct
[864]        \ifx\hl@opt@bstyle \empty
[865]          \ifx\hl@opt@bwidth \empty
[866]            \ifx\hl@opt@bdash \empty
[867]              \let\hl@BSspec\empty % don't construct
[868]            \fi
[869]          \fi
[870]        \fi
[871]        % Construct common preamble of a link.
[872]        \def\hlstart@preamble{%
[873]          \pdfstartlink
[874]            \ifx\hl@opt@width  \empty \else width  \hl@opt@width  \fi
[875]            \ifx\hl@opt@height \empty \else height \hl@opt@height \fi
[876]            \ifx\hl@opt@depth  \empty \else depth  \hl@opt@depth \fi
[877]            attr{%
[878]              \ifx\hl@opt@bcolor\empty\else /C[\hl@opt@bcolor]\fi
[879]              \ifx\hl@opt@hlight\empty\else /H/\hl@opt@hlight\fi
[880]              \ifx\hl@BSspec\relax
[881]                /BS<<%
[882]                  /Type/Border%
[883]                  \ifx\hl@opt@bstyle\empty\else /S/\hl@opt@bstyle\fi
[884]                  \ifx\hl@opt@bwidth\empty\else /W \hl@opt@bwidth\fi
[885]                  \ifx\hl@opt@bdash\empty \else /D[\hl@opt@bdash]\fi
[886]                >>%
[887]              \fi
[888]            }%
[889]        }%
[890]        % Call the handler.
[891]        \csname hl@typeh@\@hltype\endcsname
[892]      \fi
[893]    }%
[894]    % Define handlers for each supported link type (this also tells
[895]    % \hl@getparam what types we support).
[896]    \let\hl@typeh@raw\empty
[897]    \def\hl@typeh@name{\hlstart@preamble goto name{\@hllabel}}%
```

```
[898]    \def\hl@typeh@num{\hlstart@preamble  goto num \@hllabel}%
[899]    \def\hl@typeh@page{%
[900]      % PDF requires pages to start from 0, so adjust page number.
[901]      \count@=\@hllabel
[902]      \advance\count@ by-1
[903]      %
[904]      \hlstart@preamble
[905]      user{%
[906]        /Subtype/Link%
[907]        /Dest%
[908]          [\the\count@
[909]            \ifx\hl@opt@pagefit\empty/Fit\else\hl@opt@pagefit\fi]%
[910]      }%
[911]    }%
[912]    \def\hl@typeh@filename{\hl@file{(\@hllabel)}}%
[913]    \def\hl@typeh@filepage{%
[914]      % PDF requires pages to start from 0, so adjust page number.
[915]      \count@=\@hllabel
[916]      \advance\count@ by-1
[917]      %
[918]      \hl@file{%
[919]        [\the\count@ \ifx\hl@opt@pagefit\empty/Fit\else\hl@opt@pagefit\fi]%
[920]      }%
[921]    }%
[922]    \def\hl@file##1{%
[923]      \hlstart@preamble
[924]      user{%
[925]        /Subtype/Link%
[926]        /A<<%
[927]          /Type/Action%
[928]          /S/GoToR%
[929]          /D##1%
[930]          /F(\hl@opt@file)%
[931]          \ifx\hl@opt@newwin\empty \else
[932]            /NewWindow \ifcase\hl@opt@newwin false\else true\fi
[933]          \fi
[934]        >>%
[935]      }%
[936]    }%
[937]    \def\hl@typeh@url{%
[938]      \hlstart@preamble
[939]      user{%
[940]        /Subtype/Link%
[941]        /A<<%
[942]          /Type/Action%
[943]          /S/URI%
[944]          /URI(\@hllabel)%
[945]        >>%
[946]      }%
[947]    }%
[948]    %
[949]    \def\@hlend{\pdfendlink\endgroup}% The group from the \@hlstart.
[950]  %
[951]  \else % PDF output is not enabled.
[952]    \message{Eplain warning: `pdftex' hyperlink driver: PDF output is^^J
[953]             \space not enabled, falling back on `nolinks' driver.}%
[954]    \hldriver@nolinks
[955]  \fi
```

```
}%
%
% Driver 'dvipdfm'.
%
\def\hldriver@dvipdfm{%
  %
  % Hyperlink destinations.
  %
  % Default type.
  \def\hldest@type{xyz}%
  % Set defaults for the options (this also tells \hl@set@opt what
  % options we support).  (We do not define \hldest@opt@raise,
  % \hl@opt@colormodel and \hl@opt@color, they are defined and used
  % outside the drivers.)
  \let\hldest@opt@left   \empty
  \let\hldest@opt@top    \empty
  \let\hldest@opt@right  \empty
  \let\hldest@opt@bottom \empty
  \let\hldest@opt@zoom   \empty
  \let\hldest@opt@cmd    \empty
  % Multiplexer for all supported destination types.
  \def\hldest@driver{%
    % Special case for 'raw' destinations.
    \ifx\@hltype\hl@raw@word
      \csname \hldest@opt@cmd \endcsname
    \else
      % Construct common preamble of a destination.
      \def\hldest@preamble{%
        pdf: dest (\@hllabel) [@thispage
      }%
      % Call the handler.
      \csname hldest@typeh@\@hltype\endcsname
    \fi
  }%
  % Define handlers for each supported destination type (this also
  % tells \hl@getparam what types we support).
  \let\hldest@typeh@raw\empty
  \def\hldest@typeh@fit{%
    \special{\hldest@preamble /Fit]}%
  }%
  \def\hldest@typeh@fith{%
    \special{\hldest@preamble /FitH
      \ifx\hldest@opt@top\empty @ypos \else \hldest@opt@top \fi]}%
  }%
  \def\hldest@typeh@fitv{%
    \special{\hldest@preamble /FitV
      \ifx\hldest@opt@left\empty @xpos \else \hldest@opt@left \fi]}%
  }%
  \def\hldest@typeh@fitb{%
    \special{\hldest@preamble /FitB]}%
  }%
  \def\hldest@typeh@fitbh{%
    \special{\hldest@preamble /FitBH
      \ifx\hldest@opt@top\empty @ypos \else \hldest@opt@top \fi]}%
  }%
  \def\hldest@typeh@fitbv{%
    \special{\hldest@preamble /FitBV
      \ifx\hldest@opt@left\empty @xpos \else \hldest@opt@left \fi]}%
```

```
[1014]    }%
[1015]    \def\hldest@typeh@fitr{%
[1016]      \special{\hldest@preamble /FitR
[1017]        \ifx\hldest@opt@left\empty @xpos\else\hldest@opt@left\fi\space
[1018]        \ifx\hldest@opt@bottom\empty @ypos\else\hldest@opt@bottom\fi\space
[1019]        \ifx\hldest@opt@right\empty @xpos\else\hldest@opt@right\fi\space
[1020]        \ifx\hldest@opt@top\empty @ypos\else\hldest@opt@top \fi]}%
[1021]    }%
[1022]    \def\hldest@typeh@xyz{%
[1023]      \begingroup
[1024]        % Convert zoom factor:  12345 -> 12.345
[1025]        \ifx\hldest@opt@zoom\empty
[1026]          \count1=\z@ \count2=\z@
[1027]        \else
[1028]          \count2=\hldest@opt@zoom
[1029]          \count1=\count2 \divide\count1 by 1000
[1030]          \count3=\count1 \multiply\count3 by 1000
[1031]          \advance\count2 by -\count3
[1032]        \fi
[1033]        \special{\hldest@preamble /XYZ
[1034]          \ifx\hldest@opt@left\empty @xpos\else\hldest@opt@left\fi\space
[1035]          \ifx\hldest@opt@top\empty @ypos\else\hldest@opt@top\fi\space
[1036]          \the\count1.\the\count2]}%
[1037]      \endgroup
[1038]    }%
[1039]    %
[1040]    % Hyperlinks.
[1041]    %
[1042]    % Default type.
[1043]    \def\hl@type{name}%
[1044]    % We support 'url' hyperlinks, so set some group types.
[1045]    \ifx\hl@type@url\empty
[1046]      \def\hl@type@url{url}%
[1047]    \fi
[1048]    \ifx\hl@type@hrefext\empty
[1049]      \def\hl@type@hrefext{url}%
[1050]    \fi
[1051]    % Set defaults for the options (this also tells \hl@set@opt what
[1052]    % options we support).
[1053]    \def\hl@opt@bstyle  {S}%
[1054]    \def\hl@opt@bwidth  {1}%
[1055]    \let\hl@opt@bcolor  \empty
[1056]    \let\hl@opt@hlight  \empty
[1057]    \let\hl@opt@bdash   \empty
[1058]    \let\hl@opt@pagefit \empty
[1059]    \let\hl@opt@cmd     \empty
[1060]    \let\hl@opt@file    \empty
[1061]    \let\hl@opt@newwin  \empty
[1062]    % Multiplexer for all supported link types.
[1063]    \def\hl@driver{%
[1064]      % Special case for 'raw' links.
[1065]      \ifx\@hltype\hl@raw@word
[1066]        \csname \hl@opt@cmd \endcsname
[1067]      \else
[1068]        % See if we will construct a /BS spec.  We want to bother only
[1069]        % if any of \hl@opt@bstyle, \hl@opt@bwidth and \hl@opt@bdash is
[1070]        % not empty.
[1071]        \let\hl@BSspec\relax % construct
```

```
[1072]        \ifx\hl@opt@bstyle \empty
[1073]          \ifx\hl@opt@bwidth \empty
[1074]            \ifx\hl@opt@bdash \empty
[1075]              \let\hl@BSspec\empty % don't construct
[1076]            \fi
[1077]          \fi
[1078]        \fi
[1079]        % Construct common preamble of a link.
[1080]        \def\hlstart@preamble{%
[1081]          pdf: beginann
[1082]            <<%
[1083]              /Type/Annot%
[1084]              /Subtype/Link%
[1085]              \ifx\hl@opt@bcolor\empty\else /C[\hl@opt@bcolor]\fi
[1086]              \ifx\hl@opt@hlight\empty\else /H/\hl@opt@hlight\fi
[1087]              \ifx\hl@BSspec\relax
[1088]                /BS<<%
[1089]                  /Type/Border%
[1090]                  \ifx\hl@opt@bstyle\empty\else /S/\hl@opt@bstyle\fi
[1091]                  \ifx\hl@opt@bwidth\empty\else /W \hl@opt@bwidth\fi
[1092]                  \ifx\hl@opt@bdash\empty \else /D[\hl@opt@bdash]\fi
[1093]                >>%
[1094]              \fi
[1095]        }%
[1096]        % Call the handler.
[1097]        \csname hl@typeh@\@hltype\endcsname
[1098]      \fi
[1099]    }%
[1100]    % Define handlers for each supported link type (this also tells
[1101]    % \hl@getparam what types we support).
[1102]    \let\hl@typeh@raw\empty
[1103]    \def\hl@typeh@name{\special{\hlstart@preamble /Dest(\@hllabel)>>}}%
[1104]    \def\hl@typeh@page{%
[1105]      % PDF requires pages to start from 0, so adjust page number.
[1106]      \count@=\@hllabel
[1107]      \advance\count@ by-1
[1108]      %
[1109]      \special{%
[1110]        \hlstart@preamble
[1111]        /Dest[\the\count@
[1112]              \ifx\hl@opt@pagefit\empty/Fit\else\hl@opt@pagefit\fi]%
[1113]       >>%
[1114]      }%
[1115]    }%
[1116]    \def\hl@typeh@filename{\hl@file{(\@hllabel)}}%
[1117]    \def\hl@typeh@filepage{%
[1118]      % PDF requires pages to start from 0, so adjust page number.
[1119]      \count@=\@hllabel
[1120]      \advance\count@ by-1
[1121]      %
[1122]      \hl@file{%
[1123]        [\the\count@ \ifx\hl@opt@pagefit\empty/Fit\else\hl@opt@pagefit\fi]%
[1124]      }%
[1125]    }%
[1126]    \def\hl@file##1{%
[1127]      \special{%
[1128]        \hlstart@preamble
[1129]        /A<<%
```

```
[1130]          /Type/Action%
[1131]          /S/GoToR%
[1132]          /D##1%
[1133]          /F(\hl@opt@file)%
[1134]          \ifx\hl@opt@newwin\empty \else
[1135]            /NewWindow \ifcase\hl@opt@newwin false\else true\fi
[1136]          \fi
[1137]        >>%
[1138]       >>%
[1139]      }%
[1140]    }%
[1141]    \def\hl@typeh@url{%
[1142]      \special{%
[1143]        \hlstart@preamble
[1144]        /A<<%
[1145]          /Type/Action%
[1146]          /S/URI%
[1147]          /URI(\@hllabel)%
[1148]        >>%
[1149]       >>%
[1150]      }%
[1151]    }%
[1152]    %
[1153]    \def\@hlend{\special{pdf: endann}\endgroup}% The group from \@hlstart.
[1154]  }%
```

```
%
% Miscellaneous hyperlink macros.
%
%
% \href{URL}{TEXT} typesets TEXT as a link to the URL.  If URL starts
% with a #, the rest of the URL is assumed to be this document's local
% anchor.  Special chars (like # and ~) in URL don't need to be
% escaped in any way.
\def\href{%
  % Read #1 (URL) inside a group with \uncatcodespecials, to get the #
  % and ~ right.
  \bgroup
    \uncatcodespecials
    \catcode`\{=1 \catcode`\}=2
    \@href
}%
%
\def\@href#1{% We'll read #2 (TEXT) later.
  \egroup
  \edef\@hreftmp{\ifempty{#1}{}\fi}% Parameter stuffing for \@@href.
  \expandafter\@@href\@hreftmp#1\@@
}%
%
{\catcode`\#=\other
\gdef\@hrefhash{#}}%
%
\def\href@end@int{\hlend@impl{hrefint}}%
\def\href@end@ext{\hlend@impl{hrefext}}%
% Split out the first token and check if it is a #.
\def\@@href#1#2\@@{%
  \def\@hreftmp{#1}%
  \ifx\@hreftmp\@hrefhash
    \let\href@end\href@end@int
    \hlstart@impl{hrefint}{#2}%
  \else
    \let\href@end\href@end@ext
    \hlstart@impl{hrefext}{#1#2}%
  \fi
  \@@@href
}%
% Now some tricks to avoid reading the TEXT as an argument (from the
% \footnote definition in plain TeX).
\def\@@@href{%
  \futurelet\@hreftmp\href@
}%
%
\def\href@{%
  \ifcat\bgroup\noexpand\@hreftmp
    \let\@hreftmp\href@@
  \else
    \let\@hreftmp\href@@@
  \fi
  \@hreftmp
}%
%
\def\href@@{\bgroup\aftergroup\href@end \let\@hreftmp}%
%
\def\href@@@#1{#1\href@end}%
```

```
%
% Make all user-visible \hl* macros to give errors until hyperlinks
% are explicitly enabled with \enablehyperlinks.
\def\hldeston{\errmessage{Please enable hyperlinks with
  \string\enablehyperlinks\space before using hyperlink commands
  (consider selecting the 'nolinks' driver to ignore all hyperlink
  commands in your document)}}%
\let\hldestoff\hldeston \let\hlon\hldeston \let\hloff\hldeston
\let\hlstart\hldeston \let\hlend\hldeston \let\hldest\hldeston
% This catches \hltype, \hlopts, \hldesttype, \hldestopts.
\let\hl@setparam\hldeston
% Turn off all groups to make sure \hlstart@impl, \hlend@impl and
% \hldest@impl do not call \hlstart, \hlend and \hldest until
% hyperlinks are enabled.
\@hloff[*]\@hldestoff[*]%
```