

Register diagrams with field descriptions

Matthew Lovell
lovells@gmail.com

2020/03/22

Abstract

The `register` package is designed for typesetting the programmable elements in digital hardware, i.e., registers. Such registers typically have many fields and can be quite wide; they are thus a challenge to typeset in a consistent manner. This package attempts to address such issues. It is similar in some aspects to the `bitfield` package by Reuben Thomas and the `bytefield` package by Scott Pakin. Authors may also wish to inspect the `bitpattern` package by Jean-Marc Bourguet.

List of Registers

2.1	Diagnostic Control	2
2.2	Function Class	3
3.1	Example	4
3.2	Configuration	7
3.3	Color Example	9

1 Introduction

My group at work designed the memory and I/O controllers for servers and workstations. Historically, our chip documentation was done with FrameMaker or Microsoft Word. While these approaches have various disadvantages, one of the most egregious was register documentation.

The recent chips have 64-bit wide control-status registers (CSR) or, more simply, registers. Add the fact that many of these registers have a large number of single-bit fields, and you get a typesetting challenge. The typical solution was to describe such registers using a table, typing field names vertically if space became a problem. For a complicated register, these tables became quite complex and filled a large portion of a page.

When we decided to evaluate L^AT_EX for documentation, we had three goals in mind with respect to registers:

1. Create a method of documenting registers which was consistent and easy to read, regardless of the number of fields within a register.

2. Automate the creation of lists of registers, both in order of appearance within the text and in memory address order.
3. Enable the automatic extraction of documented register reset values in order to verify register functionality in the chip itself.

The `register` package is my attempt at meeting all three goals. It was first put into use in April 1999; it may not be pretty to all eyes, but it certainly has proven itself stable.

In order to promote L^AT_EX within our group at the time, we also adopted LyX. The `register` package thus attempts to work well within that environment. All LyX-specific code, however, is controlled via package options.

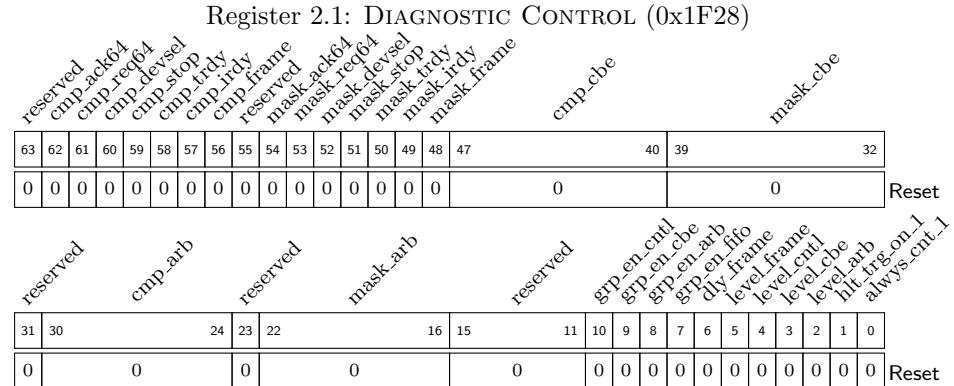
1.1 Feedback

As `register` is my first L^AT_EX package, I would welcome feedback regarding problems, new features, or more proper means of implementing existing features. For example, is there an “official” method for determining the current font’s maximum height and depth (see Section 6.4.1 for details)? I would also appreciate knowing whether anyone ever actually uses it.

It might also be worthwhile to investigate combining `register` with `bitfield` or `bytefield`.

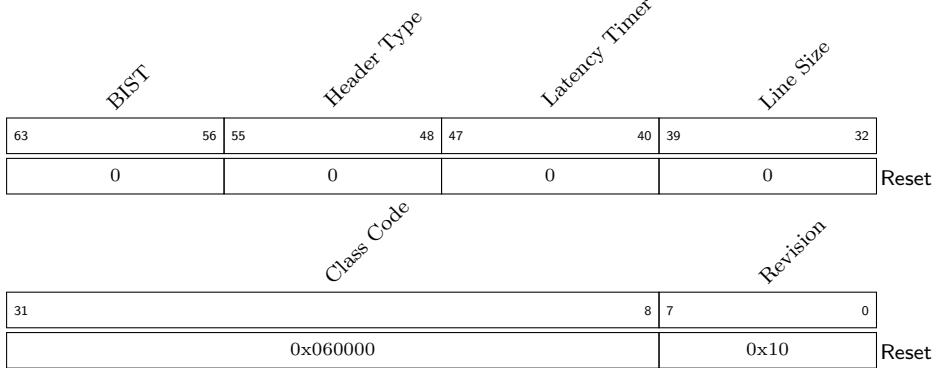
2 Examples

Register 2.1 depicts a register with many single-bit fields. Registers such as this stress any scheme of typesetting, since the field names are so much longer than the space available for them within the diagram. By typesetting the field names at an angle, `register` manages to be consistent regardless of the number or width of fields. This rotation, however, means that documents which use `register` must be produced using PostScript.



Register 2.2 is an example of a register one could derive from the PCI specification. It is a miscellaneous register in a PCI card which controls device independent functions. Note the field descriptions which, in this case, are actually part of the float.

Register 2.2: FUNCTION CLASS (0x0008)



- BIST** (Read only) Always returns 0.
- Header Type** (Read only) Always returns 0.
- Latency Timer** PCI Latency Timer value (PCI 2.2 spec, Section 6.2.4).
- Line Size** PCI Cache Line Size (PCI 2.2 spec, Section 6.2.4). Valid values are listed below; any other value will be treated as indicating 64 byte cache lines.
- 0010_0000** 128 bytes
- 0001_0000** 64 bytes
- Class Code** (Read only) PCI Class Code (PCI 2.2 spec, Section 6.2.1). Chip identifies itself as a Host bridge.
- Revision** (Read only) Chip revision number. Bits 4–7 provide the major revision number, and bits 0–3 provide the minor revision number.

A final example is shown in Figure 1. This example shows that the register drawing macros don't have to be used solely within a `register` float or environment. With Version 1.8, an unnumbered register environment is also available, using `register*`. Diagrams produced using the starred environment typeset just the register name, omitting the “Register” and number prefix.

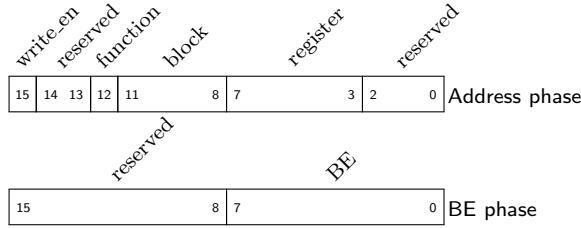
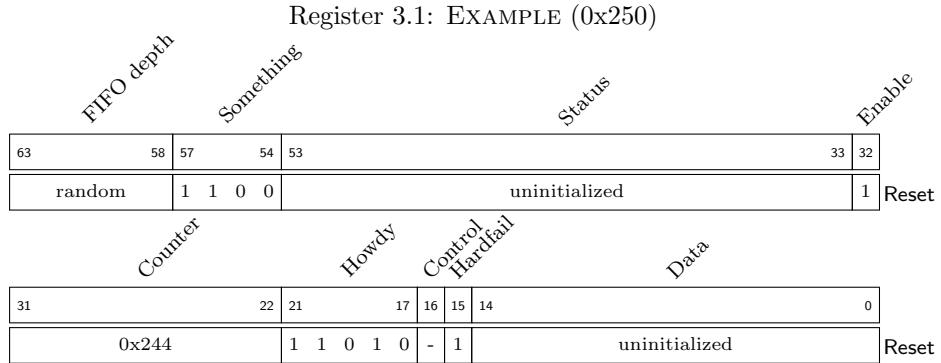


Figure 1: Address and BE phases for register access

3 User Interface

The user interface for register diagrams could be regarded as overly verbose. The main idea behind the parameters was to allow L^AT_EX *and* accompanying Perl scripts to decipher register fields and reset values. The simplest way to show the interface is via another example. Below is a 64-bit register.



It was created with the following commands:

```
\begin{register}{H}{Example}{0x250}%
\label{example}%
\regfield{FIFO depth}{6}{58}{{random}}%
\regfield{Something}{4}{54}{1100}%
\regfield{Status}{21}{33}{uninitialized}%
\regfield{Enable}{1}{32}{1}%
\reglabel{Reset}\regnewline%
\regfield{Counter}{10}{22}{0x244}%
\regfield{Howdy}{5}{17}{1_1010}%
\regfield{Control}{1}{16}{-}%
\regfield{Hardfail}{1}{15}{1}%
\regfield{Data}{15}{0}{uninitialized}%
\reglabel{Reset}\regnewline%
\end{register}
```

register The register environment begins with `\begin{register}`. The three parameters to the environment are the float specification (h, H, t, b, or p); the register name; and the register's offset in memory space. These parameters become the register caption as well as the entry in the list of registers, which is typeset using `\listofregisters`.

```
\begin{register}{\langle placement \rangle}{\langle register name \rangle}{\langle register offset \rangle}
```

\regfield Each register field is then typeset using `\regfield`. The fields must be specified¹ starting with the most-significant bit. The `\regfield` macro takes four parameters: field name, field length (in bits), starting bit number, and reset value. The reset value, unless you pass it as a single token (see example), will be spread apart to fill up the width of the field. The reset value, since it could be just a very long binary number, can be interrupted with underscores to improve readability.

```
\regfield{\langle name \rangle}{\langle length \rangle}{\langle start bit \rangle}{\langle reset value \rangle}
```

Note that the comments that are shown in the code are for the benefit of the helper scripts that accompany the `register` package. Those scripts are quite old and have *not* been updated to be tolerant of the optional color argument that is now supported. However, the general notion of using embedded comments to augment register or field information is still applicable, if one is using this package with the intent of generating HDL code from documentation.

\regBitWidth The macro `\regBitWidth` specifies the maximum number of bits to be typeset on a single line. Once a line has been completed, a new line can be inserted via `\regnewline`. Before starting a new line, however, an appropriate label for the reset value should be typed. This label is generated via `\reglabel`, which takes the string to typeset as a single parameter. The package, via the macro `\regResetName`, assumes that `Reset` will be used in order to set some lengths appropriately; it really only matters for very full lines. In a future version, it may be possible to make `\regResetName` be the default parameter to `\reglabel`.

```
\reglabel{\langle reset label \rangle}
```

As another example, consider Register 3.2 below, which has field descriptions as part of the float. This diagram was created with the following code:

```
\begin{register}{htbp}{Configuration}{0x2848}%
  \label{Configuration}%
  \regfield{soft reset perf}{1}{63}{0}%
  \regfield{reserved}{30}{33}{0}%
  \regfield{Test mode}{1}{32}{0}%
  \reglabel{Reset}\regnewline%
  \regfield{reserved}{13}{19}{0}%
  \regfield{\parbox[b]{0pt}{Request Depth}}{7}{12}{1}%

```

¹Provided you're a little-endian person!

```

\regfield{reserved}{3}{9}{0}%
\regfield{line\_2x\_L}{1}{8}{?}% STATUS
\regfield{reserved}{1}{7}{0}%
\regfield{ill\_cmd\_enable}{1}{6}{0}%
\regfield{LPCE}{1}{5}{0}%
\regfield{DVI disable}{1}{4}{0}%
\regfield{SBA enable}{1}{3}{0}%
\regfield{reserved}{2}{1}{0}%
\regfield{line\_2x\_enable}{1}{0}{0}%
\begin{regdesc}\begin{reglist}[Request~Depth]
\item [line\_2x\_enable]Setting this bit enables the chip
    to utilize a second connected data line.
\item [SBA~enable]Setting this bit activates the sideband-address port.
    The SBA port is only useable in a double-line configuration.
\item [DVI~disable]Setting this bit \emph{turns off} DVI extraction for
    DMA requests.
\item [LPCE]Line Parity Check Enable.
\item [ill\_cmd\_enable]\TR{3}Illegal Command enable.
    This bit is new for TR3.
\item [line\_2x\_L](Read only) Indicates whether this chip is connected
    to a second data line. When this bit is 0, a second line
    is available.
\item [Request~Depth]Controls number of outstanding DMA requests.
\item [Test~mode]Activates data line test mode.
\item [soft~reset~perf]\TR{4}Indicates that a soft reset has been
    performed. This bit is new for TR4.
\end{reglist}\end{regdesc}\end{register}

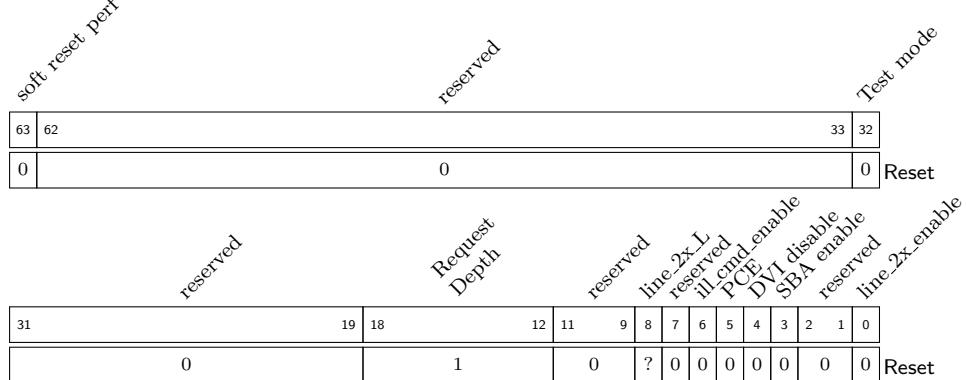
```

regdesc This example uses the `regdesc` environment, which can either be part of a register float or used standalone. The purpose of `regdesc` is really just to turn on centering and, when the LyX package option is specified, redefine the `lyxlist` environment. If one is using LyX, then register field descriptions can be entered using its List type. If one is typing L^AT_EX directly, just use the `reglist` list definition.

The `regdesc` and `reglist` environments both take optional arguments. For `regdesc`, the optional argument specifies how wide the register description body should be. The default value is 90% of `\textwidth`. For `reglist`, the optional argument is the longest field name, which allows the list to be spaced appropriately (like examples given in *The L^AT_EX Companion*).

\regfieldb Finally, if a register with no reset values is desired, one can use the `\regfieldb` macro. It takes the same first three arguments as `\regfield`. Figure 1 was produced using `\regfieldb`.

Register 3.2: CONFIGURATION (0x2848)



line_2x_enable Setting this bit enables this chip to utilize a second connected data line.

SBA enable Setting this bit activates the sideband-address port. The SBA port is only useable in a double-line configuration.

DVI disable Setting this bit *turns off* DVI extraction for DMA requests.

LPCE Line Parity Check Enable.

ill_cmd_enable Illegal OCMD enable. This bit is new for TR3.

TR3

line_2x_L (Read only) Indicates whether this chip is connected to a second data line. When this bit is 0, a second line is available.

Request Depth Controls number of outstanding DMA requests.

Test mode Activates data line test mode.

soft reset perf Indicates that a soft reset has been performed. This bit is new for TR4.

TR4

3.1 TR Flags

\TR The previous example also shows the TR flags. These marginal notes can be used to denote features that are specific to a particular release of a chip. These macros were created since L^AT_EX's \marginpar is itself a float; as such it cannot be nested inside other floats. The \TR macros thus implement something very similar to the solution for Problem 14.28 in *The T_EXbook*.

TR flags are turned on via the TRflags package option. If the option is not specified, then the \TR macro still exists but is empty. The TR flags are placed inside an \fbox if the TRboxed package option is specified.

For two-sided documents, the TR flags are placed in the outer margins. The placement decision is made by placing labels for each TR flag and then determining even/odd page numbering in a subsequent L^AT_EX run.

3.2 Background Colors

With Version 1.7, this package also supports an optional color package option. If that option is present, then xcolor is loaded and the \regfield and \regfieldb macros support an optional first argument that is used as fill color for the boxes drawn by \framebox. Internally, \colorbox is used to provide this functionality.

Repeating our example from earlier, but with some fill colors specified:

```
\begin{register}{H}{Color Example}{0x250}%
  \label{colexample}%
  \regfield[FIFO depth]{6}{58}{{random}}%
  \regfield[green!30]{Something}{4}{54}{1100}%
  \regfield[gray!20]{Status}{21}{33}{{uninitialized}}%
  \regfield[Enable]{1}{32}{1}%
  \reglabel{Reset}\regnewline%
  \regfield[Counter]{10}{22}{0x244}%
  \regfield[red!20]{Howdy}{5}{17}{1_1010}%
  \regfield[Control]{1}{16}{-}%
  \regfield[yellow!50]{Hardfail}{1}{15}{1}%
  \regfield[gray!20]{Data}{15}{0}{{uninitialized}}%
  \reglabel{Reset}\regnewline%
\end{register}
```

now yields the following:

4 Helper Scripts

A (now old) Perl module and a Perl script are included as part of the **register** package. The module augments the package by providing the ability to parse the register macro information into Perl. This information consists of the parameters used within the macros themselves, as well as the comments which follow. If you examine the last register example, you will notice that some fields specify **READ_ONLY** or **STATUS** in a comment. The “directives” can be used to indicate that particular fields are unaffected by writes or unaffected by reset, respectively.

The script, `reg_list.pl`, utilizes the Perl module to produce lists of registers in offset-order or to generate reset value and mask information for each register. The later data can be used in simulation to ensure that registers behave as documented.

Users are cautioned that the Perl script, since it is no longer in active use by the author, has not been modified to support the optional color argument for the `regfield` macro.

5 Version History

v0.1 (1999/03/29) First release

v0.2 (2000/01/29) Added regdesc environment, for use under LyX

v0.21 (2000/03/12) Fixed minute problem shown in regs with lots of 1-bit fields

v0.3 (2000/06/19) Added reglabelb macro, for setting label without a drop

v0.31 (2000/07/11) Modified names of lengths used within macros

v0.32 (2000/08/02) Modified regdesc env to use regdescsep for vertical separation

v0.4 (2001/08/11) Added macros for marginal Tape Release flags

v0.5 (2001/08/12) Added package options for handling LyX, hyperref, and TR flags

- v0.6** (2001/08/19) Moved chapter redefinition into package; improved TR flags
- v1.0** (2001/09/06) Created .dtx documentation; first public release
- v1.01** (2003/01/17) Removed automatic label in register diagrams
- v1.1** (2003/02/15) Changed length manipulation to work with parskip package
- v1.2** (2003/03/31) Removed requirement to specify register offset
- v1.3** (2003/11/24) Changed how center is handled within register environment
- v1.4** (2004/08/16) Now provide a boolean set only in register context
- v1.5** (2007/03/08) Corrected use of conditionals used in spreading register reset value
- v1.6** (2011/01/11) Mainmatter correction suggested by Kjetil Oftedal
- v1.6.1** (2018/05/19) Published with updated email address; no functional changes
- v1.7** (2018/08/18) Added color option to package, suggested by Marco Stolba
- v1.8** (2018/11/10) Added unnumbered `register*` environment, suggested and written by Stephan Bauroth
- v1.9** (2019/01/01) Switched to a re-definable macro to provide default float name and LoR name, suggested by Benyuan Liu.
- v2.0** (2020/03/22) Added package options for captions and replaced `\bf` usage, suggested by Andrea Bettati. Also attempt to work better with `memoir` class.

6 Implementation

6.1 Identification & Options

The `register` document class can only be used with L^AT_EX 2 _{ε} , so we make sure that an appropriate message is displayed when another T_EX format is used.

```
1 {*}\\NeedsTeXFormat{LaTeX2e}
2 \\ProvidesPackage{register} [2020/03/22 v2.0 Register macros with
3 hyperref/LyX support]
4
5 \\RequirePackage{ifthen}[1997/11/02]
6 \\RequirePackage{graphicx}[1997/06/09]
7 \\RequirePackage{float}[2001/07/25]
8 \\RequirePackage{calc}[1998/06/07]
```

Declare the package options and some booleans. There are options indicating whether `register` is being used with LyX or `hyperref`. The marginal TR flags can also be turned on or off with an option. A new package option, `botcaption` serves to change the default caption location to below the register diagram, rather than above. Finally, `nocaption` turns off the `\\caption` invocation internal to the `register` environment entirely, for users who would rather craft their own caption. Note than one can specify *both* `botcaption` and `nocaption` – explicit author-specified captions would then appear at the bottom of each diagram.

```
9 %
10 \\DeclareOption{LyX}{\\setboolean{RegisterLyX}{true}}
11 \\DeclareOption{hyperref}{\\setboolean{RegisterHyperref}{true}}
12 \\DeclareOption{TRflags}{\\setboolean{RegisterTRFlags}{true}}
13 \\DeclareOption{TRboxed}{\\setboolean{RegisterTRBoxed}{true}}
14 \\DeclareOption{color}{\\setboolean{RegisterColors}{true}}
15 \\DeclareOption{botcaption}{\\setboolean{RegisterBottom}{true}}
16 \\DeclareOption{nocaption}{\\setboolean{RegisterNoCaption}{true}}
17
18 \\DeclareOption*{%
19   \\PackageWarning{register}{Unknown option '\\CurrentOption'}%
20 }
21
22 \\newboolean{RegisterLyX}%
23 \\newboolean{RegisterHyperref}%
24 \\newboolean{RegisterTRFlags}%
25 \\newboolean{RegisterTRBoxed}%
26 \\newboolean{RegisterColors}%
27 \\newboolean{RegisterBottom}%
28 \\newboolean{RegisterNoCaption}%
29
30 \\setboolean{RegisterLyX}{false}
31 \\setboolean{RegisterHyperref}{false}
32 \\setboolean{RegisterTRFlags}{false}
33 \\setboolean{RegisterTRBoxed}{false}
```

```

34 \setboolean{RegisterColors}{false}
35 \setboolean{RegisterBottom}{false}
36 \setboolean{RegisterNoCaption}{false}
37
38 \ProcessOptions\relax % Process package options

```

If the `color` package option has been specified, go ahead and load the `xcolor` package as well.

```
39 \ifthenelse{\boolean{RegisterColors}}{\RequirePackage{xcolor}[2007/01/21]}{}
```

Next, provide a flag indicating when one is inside a register context. Some users may wish to define macros, for example, which make use of `\marginpar`. Those cannot be used inside of a floating register. So, this flag is provided to provide something to test against.

```

40 \newboolean{RegisterContext}
41 \setboolean{RegisterContext}{false}
```

`\regFloatName` Below are some macros that support a little internationalization, expanded upon
`\regListName` a suggestion from Benyuan Liu. The `\regFloatName` provides the default string
`\regResetName` used to label each register diagram. The string with which to label reset values
is `\regResetName`. The font size for field names and for the `\regResetName` is
controlled by `\regLabelSize`, defined further below.

```

42 \newcommand{\regFloatName}{Register}
43 \newcommand{\regListName}{List of Registers}
44 \newcommand{\regResetName}{Reset}
```

6.2 New Float Declaration

Declare the new register float type and the lengths which the macros will use. The register diagrams tend to look best with the caption at the top of the float. Note, however, that caption placement *can* be changed in the document itself. For example, one can use the macro `\floatstyle{plain}`, followed by `\restylefloat{Regfloat}`, to alter how all subsequent register captions are typeset.

```
45 \ifthenelse{\boolean{RegisterBottom}}{\floatstyle{plain}}{\floatstyle{plaintop}}
```

If `\chapter` is defined in the current document class, then the register diagrams will count by chapter, otherwise they will count by section.

```

46 \@ifundefined{chapter}
47 {\newfloat{Regfloat}{tbp}{rdf}{section}}
48 {\newfloat{Regfloat}{tbp}{rdf}{chapter}}
49 \floatname{Regfloat}{\regFloatName}
```

6.3 Style Parameters

`\regWidth` The `\regWidth` length controls how physically wide the register diagrams are,
`\regBitWidth` while `\regBitWidth` specifies how the maximum number of bits per line in the

diagram. The default value for `\regWidth` is 95% of the width of the main text body.

```

\regBitSize The \regBitSize command controls the font size used for bit values, \regResetSize
\regBitFamily controls the font size used for reset values, and \regBitFamily controls which font
\regResetSize family is used to denote bit positions. Finally, \regDescFamily controls the ap-
\regLabelSizepearance of field names within register descriptions.

\regLabelFamily
\regDescFamily 50 \newlength{\regWidth}
51 \newlength{\regFieldLen}
52 \newlength{\regLabelAdjust}
53 \newlength{\regResetHeight}
54 \newlength{\regResetDepth}
55 \newlength{\regResetDrop}
56 \newlength{\regDescSkip}
57 \newlength{\regRsvdHeight}
58 \newlength{\regRsvdDrop}
59 \newlength{\regFboxSep}
60 \setlength{\regWidth}{0.95\textwidth}
61 \newcommand{\regBitWidth}{32}
62 \newcommand{\regBitSize}{\tiny}
63 \newcommand{\regBitFamily}{\sffamily}
64 \newcommand{\regResetSize}{\scriptsize}
65 \newcommand{\regLabelSize}{\footnotesize}
66 \newcommand{\regLabelFamily}{\rmfamily}
67 \newcommand{\regDescFamily}{\bfseries}

```

The lengths below control the vertical spacing between a register diagrams and the register field descriptions. This distance must change when the `\regdesc` is part of the `\Regfloat` or standalone.

```

68 \newlength{\regdescsep}
69 \newlength{\oldregdescsep}
70 \setlength{\regdescsep}{-\medskipamount}
71
72 \newsavebox{\Label}
73 \newsavebox{\RotatedLabel}
74 \newcounter{upperbit}
75 \newcounter{lowerbit}

```

6.4 Register macros

`\reglist` Define a new environment for register field descriptions, beneath a register diagram. These descriptions can indicate what a particular field is used for, whether it is read-only, etc. The field descriptions themselves are in a `\reglist` list environment, unless LyX is being used.

```

76 \newenvironment{reglist}[1][M]
77   {\begin{list}{}}
78     {\settowidth{\labelwidth}{\regDescFamily #1}}
79     \addtolength{\labelwidth}{\labelsep}

```

```

80      \setlength{\leftmargin}{\labelwidth}
81      \addtolength{\leftmargin}{\labelsep}
82      \addtolength{\leftmargin}{0.5\regDescSkip}
83      \addtolength{\rightmargin}{0.5\regDescSkip}
84      \setlength{\topsep}{0pt}
85      \setlength{\itemsep}{0pt}
86      \setlength{\parsep}{0.5\baselineskip}
87      \renewcommand{\makelabel}[1]{\regDescFamily ##1 \hfill}}
88  {\end{list}}

```

\regdesc Next is the `\regdesc` environment. If LyX is being used, the environment temporarily redefines the `lyxlist` environment, which is what LyX provides as the default list structure. Otherwise, the user is left to use `\reglist`. The redefinition should look very similar to that for `\reglist`.

In all cases, `\regdesc` starts by attempting to place a small amount of vertical space between it and the presumed register diagram immediately above it.

```

89 \newenvironment{regdesc}[1][0.90\textwidth]%
90 % -----
91 {%
92   \setlength{\regDescSkip}{\textwidth - #1}%
93   \vspace{\regdescsep}%
94 \ifthenelse{\boolean{RegisterLyX}}{%
95   \renewenvironment{lyxlist}[1]
96   {\begin{list}{}%
97    \settowidth{\labelwidth}{\regDescFamily ##1}
98    \addtolength{\labelwidth}{\labelsep}
99    \setlength{\leftmargin}{\labelwidth}
100   \addtolength{\leftmargin}{\labelsep}
101   \addtolength{\leftmargin}{0.5\regDescSkip}
102   \addtolength{\rightmargin}{0.5\regDescSkip}
103   \setlength{\topsep}{0pt}
104   \setlength{\partopsep}{0pt}
105   \renewcommand{\makelabel}[1]{\regDescFamily ####1 \hfill}}%
106   {\end{list}}}{}%
107 % endif
108 % set spacing appropriately
109 \leftskip 0.5\regDescSkip%
110 \rightskip 0.5\regDescSkip%
111 \parfillskip=\z@ plus 1fil%
112 \parskip=0.5\baselineskip \advance\parskip by 0pt plus 2pt%
113 }% end begin{regdesc}
114 % -----
115 {\vskip\baselineskip}

```

\regnewline This command allows a register diagram to be broken into multiple lines

```
116 \newcommand{\regnewline}{\\*}
```

\register This environment declares the floating environment in which a register diagram

and, optionally, its field descriptions can be typeset. It takes as arguments the float type (h, t, p, H); the register name; and the register offset/address.

```

117 \newenvironment{register}[3]
118 {\begin{Regfloat}[\#1]%
119   \setlength{\leftskip}{0pt}%
120   \setlength{\oldregdescsep}{\regdescsep}%
121   \setlength{\regdescsep}{0.5\baselineskip}%
122   \setlength{\partopsep}{0pt}%
123   \setlength{\topsep}{0pt}%
124   \setboolean{RegisterContext}{true}%
125   \ifthenelse{\boolean{RegisterNoCaption}}{}{%
126     \ifthenelse{\equal{\#3}{}}{%
127       {\caption[\#2]{\textsc{\#2}}}%
128     }{%
129       {\caption[\#2]{\textsc{\#2} (\#3)}}%
130     }%
131   }%
132   \centering%
133   % restore lengths%
134   \leftskip\z@%
135   \rightskip\z@%
136   \parfillskip=\z@ plus 1fil%
137   \setlength{\regdescsep}{\oldregdescsep}%
138   \setboolean{RegisterContext}{false}%
139 \end{Regfloat}}

```

\register* This register* environment provides for an unnumbered or “captionless” register; registers in such an environment are not added to the list of registers.

```

138 \newenvironment{register*}[3]
139 {\begin{Regfloat}[\#1]%
140   \setlength{\leftskip}{0pt}%
141   \setlength{\oldregdescsep}{\regdescsep}%
142   \setlength{\regdescsep}{0.5\baselineskip}%
143   \setlength{\partopsep}{0pt}%
144   \setlength{\topsep}{0pt}%
145   \setboolean{RegisterContext}{true}%
146   \ifthenelse{\equal{\#3}{}}{%
147     \centering\textsc{\#2}\%%
148   }{%
149     \centering\textsc{\#2} (\#3)\%%
150   }%
151   \centering%
152   % restore lengths%
153   \leftskip\z@%
154   \rightskip\z@%
155   \parfillskip=\z@ plus 1fil%
156   \setlength{\regdescsep}{\oldregdescsep}%
157   \setboolean{RegisterContext}{false}%
158 \end{Regfloat}}

```

\regUnderScore The next definitions provide a method of spreading out an argument, i.e., placing \regFiller’s between each character. They can be used to space the reset value \regSpread

of a register field appropriately. Previous versions of these macros used TeX's conditionals incorrectly.

If you are using the `underscore` package, you should *repeat* the definition of `\regUnderScore`, after loading `underscore` in order to observe the changes made by that package. The spreading macros below strive to strip out underscores.

```

157 \def\regUnderScore{_}%
158 \def\regFiller#1{\def\regInner{#1}%
159 \ifx\regInner\regUnderScore%
160 \else%
161 \ifnum\count0>0%
162 \hfill#1%
163 \else#1\fi%
164 \fi%
165 \advance\count0 by 1%
166 }%
167 \def\regSpread#1{\count0=0{}\regSpreadaux#1\empty}%
168 \def\regSpreadaux#1#2\empty{\def\aux{#1}\show#1%
169 \ifx\aux\empty%
170 \else%
171 \def\aux{#2}%
172 \regFiller{\aux}%
173 \ifx\aux\empty%
174 \else%
175 \regSpreadaux#2\empty%
176 \fi\fi}

```

6.4.1 Register fields

Define some internal utility macros which get called repeatedly. These macros figure out some of the dimensions of the current font (is there a better way to do this?), construct and rotate label boxes, and typeset bit positions.

```

177 \newcommand{\setRegLengths}{%
178   % Compute basic width of a single bit
179   \settowidth{\regFieldLen}{\regLabelSize \regResetName}%
180   \setlength{\regFieldLen}{\regWidth - \regFieldLen}%
181   \setlength{\regFieldLen}{\regFieldLen / \regBitWidth}%
182   % Figure out height and depth of reset field in current font
183   % Is there a more ‘‘official’’ method to do this?
184   \settodepth{\regResetDepth}{\regResetSize ()jgpq}%
185   \settoheight{\regResetHeight}{\regResetSize ()ABCjkl}%
186   \addtolength{\regResetHeight}{\regResetDepth}%
187   % Compute how far to drop the reset fields down. The value at
188   % the end is effectively the separation between the bit position
189   % box and the reset value box.
190   \setlength{\regResetDrop}{\regResetHeight + 2\fboxsep - 2\fboxrule + 3pt}%
191   % New lengths to support colorbox use, when fboxsep gets set to 0
192   \setlength{\regRsvdDrop}{\regResetDrop + \fboxsep}%
193   \setlength{\regRsvdHeight}{\regResetHeight + 2\fboxsep}%

```

```

194 \setlength{\regFboxSep}{\fboxsep}%
195 }

```

These macros assembly, rotate, and typset the register field name. It is the use of `\rotatebox` below which makes `register` require PostScript processing.

```

196 \newcommand{\regMakeFieldName}[1]{%
197 % Create box to hold label
198 \savebox{\Label}{\regLabelSize\regLabelFamily #1}%
199 }
200 \newcommand{\regRotateFieldName}{%
201 \savebox{\RotatedLabel}{\rotatebox[origin=LB]{45}{\usebox{\Label}}}%
202 \makebox[0pt][1]{\raisebox{\regResetHeight + \fboxsep + \depth + 1pt}{%
203 \hspace{\regLabelAdjust}\usebox{\RotatedLabel}}}%
204 }

```

The `\typesetRegBits` macro constructs the frame for a particular register field and sets the starting and ending bit positions within that frame.

```

205 \newcommand{\typesetRegBits}[1]{%
206 \ifthenelse{#1 > 1}{%
207 {\framebox[\regFieldLen][c]{%
208 \regBitSize \rule[-1\regResetDepth]{0pt}{\regResetHeight}%
209 \regBitFamily\arabic{upperbit} \hfill \arabic{lowerbit}}}%
210 {\framebox[\regFieldLen][c]{%
211 \regBitSize \rule[-1\regResetDepth]{0pt}{\regResetHeight}%
212 \regBitFamily\arabic{lowerbit}}}%
213 }

```

This macro constructs a frame, below the bit position frame, displaying the reset (or some other) value for a register field.

```

214 \newcommand{\typesetRegReset}[1]{%
215 % Typeset reset value in a framebox
216 \makebox[0pt][1]{\raisebox{-1\regResetDrop}{\framebox[\regFieldLen][c]{%
217 % Place an invisible rule to control the box
218 % surrounding the reset field
219 \regResetSize \rule[-1\regResetDepth]{0pt}{\regResetHeight}\regSpread{#1}}}}%
220 }

```

`\regfieldNoColor` The macros below are the user interface to the register drawing routines. The first of these, `\regfieldNoColor` takes four arguments: the field name, field length (in bits), the starting bit number, and the field's reset value. This is the version of the macro that does not make any use of `\colorbox`.

```

221 \newcommand{\regfieldNoColor}[4]{%
222 % Compute overall field length
223 \setRegLengths%
224 \setlength{\regFieldLen}{#2\regFieldLen + \fboxrule}%
225 % Figure out bit positions
226 \setcounter{lowerbit}{#3}%
227 \setcounter{upperbit}{#3 + #2 - 1}%
228 \regMakeFieldName{#1}%
229 % Figure out how far over to place label, accounting for height

```

```

230 \setlength{\regLabelAdjust}{0.5\regFieldLen - 0.707107\ht\Label}%
231 % Now, rotate and type the label
232 \regRotateFieldName%
233 \typesetRegReset{#4}%
234 % Typeset bit positions in a framebox
235 \typesetRegBits{#2}%
236 \hspace{-1\fboxrule}%
237 }

```

- \regfieldbNoColor** The `\regfieldbNoColor` macro can be used to construct a register diagram without reset values. It's parameters are thus the same as for `\regfieldNoColor` with the omission of reset value.

```

238 \newcommand{\regfieldbNoColor}[3]{%
239 % Compute overall field length
240 \setRegLengths%
241 \setlength{\regFieldLen}{#2\regFieldLen + \fboxrule}%
242 % Figure out bit positions
243 \setcounter{lowerbit}{#3}%
244 \setcounter{upperbit}{#3 + #2 - 1}%
245 % Create box to hold label
246 \regMakeFieldName{#1}%
247 % Figure out how far over to place label, accounting for height
248 \setlength{\regLabelAdjust}{0.5\regFieldLen - 0.707107\ht\Label}%
249 % Now, rotate and typeset the label
250 \regRotateFieldName%
251 % Typeset bit positions
252 \typesetRegBits{#2}%
253 \hspace{-1\fboxrule}%
254 }

```

- \typesetRegColorBits** Now, we define versions of the same macros that make use of `\colorbox`, taking an additional argument that specifies the fill color to use.

```

\regfieldColor
\regfieldbColor
255 \ifthenelse{\boolean{RegisterColors}}{%
256 \newcommand{\typesetRegColorBits}[2]{%
257 \ifthenelse{#2 > 1}{%
258 {\setlength\fboxsep{0pt}%
259 \colorbox{#1}{%
260 \framebox[\regFieldLen][c]%
261 {\regBitSize \rule[-1\regResetDepth - \regFboxSep]{0pt}%
262 {\regResetHeight + 2\regFboxSep}%
263 \regBitFamily\hspace{\regFboxSep}%
264 \arabic{upperbit} \hfill \arabic{lowerbit}%
265 \hspace{\regFboxSep}}}%
266 {\setlength\fboxsep{0pt}%
267 \colorbox{#1}{%
268 \framebox[\regFieldLen][c]%
269 {\regBitSize \rule[-1\regResetDepth - \regFboxSep]{0pt}%
270 {\regResetHeight + 2\regFboxSep}}%

```

```

271           \regBitFamily{arabic}{lowerbit}}}}}}%
272   \setlength\fboxsep{\regFboxSep}%
273 }
274 \newcommand{\typesetRegColorReset}[2]{%
275 % Typeset reset value in a framebox
276   \makebox[0pt][1]{\raisebox{-1\regRsvdDrop}{%
277     \setlength\fboxsep{0pt}%
278     \colorbox{#1}{\framebox[\regFieldLen][c]{}%
279     % Place an invisible rule to control the box surrounding the reset field
280     \rule[-1\regResetDepth]{0pt}{\regRsvdHeight}%
281     \raisebox{\regFboxSep}{\makebox[\regFieldLen]{%
282       \hspace{\regFboxSep}\regSpread{#2}\hspace{\regFboxSep}}}}}}}%
283 }{%
284 \newcommand{\regfieldColor}[5]{%
285 % Compute overall field length
286   \setRegLengths%
287   \setlength{\regFieldLen}{#3\regFieldLen + \fboxrule}%
288 % Figure out bit positions
289   \setcounter{lowerbit}{#4}%
290   \setcounter{upperbit}{#4 + #3 - 1}%
291   \regMakeFieldName{#2}%
292 % Figure out how far over to place label, accounting for height
293   \setlength{\regLabelAdjust}{0.5\regFieldLen - 0.707107\ht\Label}%
294 % Now, rotate and type the label
295   \regRotateFieldName%
296   \typesetRegColorReset{#1}{#5}%
297 % Typeset bit positions in a framebox
298   \typesetRegColorBits{#1}{#3}%
299   \hspace{-1\fboxrule}%
300 }{%
301 \newcommand{\regfieldbColor}[4]{%
302 % Compute overall field length
303   \setRegLengths%
304   \setlength{\regFieldLen}{#3\regFieldLen + \fboxrule}%
305 % Figure out bit positions
306   \setcounter{lowerbit}{#4}%
307   \setcounter{upperbit}{#4 + #3 - 1}%
308 % Create box to hold label
309   \regMakeFieldName{#2}%
310 % Figure out how far over to place label, accounting for height
311   \setlength{\regLabelAdjust}{0.5\regFieldLen - 0.707107\ht\Label}%
312 % Now, rotate and typeset the label
313   \regRotateFieldName%
314 % Typeset bit positions
315   \typesetRegColorBits{#1}{#3}%
316   \hspace{-1\fboxrule}%
317 }{%
318 }{%

```

\regfield Next, provide definitions for the \regfield and \regfieldb macros that are \regfieldb

intended for direct use within the register diagrams. These commands just check for the presence of a non-empty optional argument and, based upon that, decide whether to invoke the Color or NoColor variants of the typesetting macros.

```

319 \ifthenelse{\boolean{RegisterColors}}{
320   \newcommand{\regfield}[5][]{%
321     \ifthenelse{\equal{\#1}{}}{%
322       \regfieldNoColor{\#2}{\#3}{\#4}{\#5}%
323     }{%
324       \regfieldColor{\#1}{\#2}{\#3}{\#4}{\#5}%
325     }%
326   \newcommand{\regfielddb}[4][]{%
327     \ifthenelse{\equal{\#1}{}}{%
328       \regfielddbNoColor{\#2}{\#3}{\#4}%
329     }{%
330       \regfielddbColor{\#1}{\#2}{\#3}{\#4}%
331     }%
332   \newcommand{\regfieldb}[4][]{%
333     \regfielddbNoColor{\#2}{\#3}{\#4}%
334 }
```

\regbits The `\regbits` macro typesets a register field without showing bit positions. As such, it only takes three parameters: field name, field bit length, and field value.

```

334 \newcommand{\regbits}[3]{%
335   % Compute overall field length
336   \setRegLengths%
337   \setlength{\regFieldLen}{\#2\regFieldLen + \fboxrule}%
338   % Figure out bit positions
339   \setcounter{lowerbit}{\#3}%
340   \setcounter{upperbit}{\#3 + \#2 - 1}%
341   % Create box to hold label
342   \regMakeFieldName{\#1}%
343   % Figure out how far over to place label, accounting for height
344   \setlength{\regLabelAdjust}{0.5\regFieldLen - 0.707107\ht{\Label}}%
345   % Now, rotate and typeset the label
346   \regRotateFieldName%
347   % Typeset field value
348   \framebox[\regFieldLen][c]{%
349     \tiny\regSpread{\#3}%
350   \hspace{-1\fboxrule}%
351 }
```

\regspace Finally, define some additional utility macros. An invisible box with the same width as a specified number of bits is typeset by `\regspace`. It takes as its sole parameter the desired bitwidth. The `\reglabel` macro typesets the label for the reset field in register diagrams. Finally, `\reglabelb` performs the same function, but without any drop.

```

352 \newcommand{\regspace}[1]{%
353   % Compute overall field length
```

```

354 \setRegLengths%
355 \setlength{\regFieldLen}{\regFieldLen + \fboxrule}%
356 \makebox[\regFieldLen]{}
357 }
358
359 \newcommand{\reglabel}[1]{%
360   \settowidth{\regFieldLen}{\regLabelSize \regResetName}%
361   \setlength{\regResetDrop}{\regResetDrop + 0.5\fboxsep}%
362   $\\,$\raisebox{-1\regResetDrop}{\makebox[\regFieldLen][1]%
363     {\regLabelSize\regBitFamily #1}}%
364 }
365
366 \newcommand{\reglabelb}[1]{%
367   \settowidth{\regFieldLen}{\regLabelSize \regResetName}%
368   $\\,$\raisebox{-0.5\fboxsep}{\makebox[\regFieldLen][1]%
369     {\regLabelSize\regBitFamily #1}}%
370 }

```

6.4.2 Hyperref Interface

This code helps with the hyperlinks to register floats when producing linked PDF.

```

371 \ifthenelse{\boolean{RegisterHyperref}}{%
372   % Define a counter for the hyperref package. Otherwise,
373   % the hyperlinks to registers don't work correctly
374   \cnamedef{theHRegfloat}{\theRegfloat}
375
376   % Define a bookmark level for Regfloats (for hyperref package)
377   \def\toclevel@Regfloat{0}
378 }{}

```

6.4.3 List of Registers

The code below redefines what the `float` package specifies in order to provide a hook for changing the format of the ToC line for registers.

```

379 \newcommand{\listofregisters}{%
380   \@ifundefined{ext@Regfloat}{\float@error{Regfloat}}{%
381     \@ifundefined{chapter}{\def\@tempa{\section*}}{%
382       \def\@tempa{\chapter*}}%
383     \@tempa{\regListName\@mkboth{\uppercase{\regListName}}{%
384       \uppercase{\regListName}}}%
385     \addcontentsline{toc}{chapter}{\regListName}%
386     \starttoc{\@nameuse{ext@Regfloat}}}}%
387 \newcommand{\@Regfloat{\dottedtocline{1}{1.5em}{2.3em}}}

```

The `\chapter` command, if present, is also redefined in order to get identical chapter-related spacing in the list of registers as appears in the LoF and LoT. Kjetil Oftedal provided a fix wherein we test that `@mainmatter` is defined prior to using it.

This change to `\chapter` is not particularly friendly if one is using the `memoir` package, however! So, we first check what document class is being used. If `memoir` is being used, we invoke its `\memchapinfo` hook rather than attempt any redefinition of `\chapter`.

```

388 \@ifundefined{@mainmatter}{\newif\if@mainmatter \@mainmattertrue{}}
389 \@ifclassloaded{memoir}{%
390 \renewcommand{\memchapinfo}[4]{%
391   \addtocontents{rdf}{\protect\addvspace{10pt}}}
392 }{%
393 \@ifundefined{chapter}{}%
394 { % Adjust chapter definition slightly for Regfloats
395 \def\@chapter[#1]#2{\ifnum \c@sectiondepth >\m@ne
396   \if@mainmatter
397     \refstepcounter{chapter}%
398     \typeout{@chapapp\space\thechapter.}%
399     \addcontentsline{toc}{chapter}%
400     {\protect\numberline{\thechapter}#1}%
401   \else
402     \addcontentsline{toc}{chapter}{#1}%
403   \fi
404   \else
405     \addcontentsline{toc}{chapter}{#1}%
406   \fi
407   \chaptermark{#1}%
408   \addtocontents{lof}{\protect\addvspace{10\p@}}%
409   \addtocontents{lot}{\protect\addvspace{10\p@}}%
410   \addtocontents{rdf}{\protect\addvspace{10\p@}}% --- Add space ---
411   \if@twocolumn
412     \atopnewpage[\@makechapterhead{#2}]%
413   \else
414     \@makechapterhead{#2}%
415   \fi
416 }%
417 }%

```

6.5 TR flag macros

Define a mechanism for including TR flags in register descriptions, as well as possibly in the main body of the text. One cannot just use `\marginpar`'s within a register description, since they are typically part of a register float. L^AT_EX's `\marginpar` command is *also* considered a float, so they cannot be nested.

So, we resort to using some lower level L^AT_EX. This certainly isn't the most elegant way to get the flags, but I think it will work for the majority of cases.

The `\TR{x}` command is intended to be placed at the start of a register field description, so that the alignment is obvious on the page. To also get decent alignment in main body text, it is necessary to have a second definition. So, we'll see these commands get redefined inside `regdesc`, below.

The TR flags also decide which page they are on by placing labels and testing

the resulting pagenumber. This enables the flags to be placed in the outer margin in two-sided documents.

\TRfamily The font family used for the TR flags can be controlled via \TRfamily. The \TRwidth width available for TR flags is controlled by \TRwidth, which by default is set to \marginparwidth. Setting the width may be important if you are using the TRboxed package option.

```

418 % Define a pageref which will work with \isodd
419 \newcommand{\GetTRSecondParam}{}
420 \long\def{\GetTRSecondParam#1#2#3}{\nil{#2}}
421 \newcommand*{\GetTRPageRef}[1]{%
422     \expandafter\expandafter\expandafter\GetTRSecondParam
423     \csname r@#1\endcsname
424     % dummy, if the reference is undefined
425     \nil
426 }
427
428 \newcommand{\TRfamily}{\sffamily}
429 \newcounter{T@peReleaseTag}
430 \newlength{\TRwidth}
431 \setlength{\TRwidth}{\marginparwidth}
432 \newlength{\T@peReleaseDepth}
433
434 % Internal command to form the actual TR margin note.
435 \newcommand{\TRwriteout}[1]{%
436     \makebox[\TRwidth][c]{%
437         \raisebox{\T@peReleaseDepth}{%
438             \ifthenelse{\boolean{RegisterTRBoxed}}{%
439                 {\fbox{\TRfamily #1}}{%
440                 {\TRfamily #1}}}}%
441     }
442
443 % Internal command to typeset a TR flag in the right margin
444 \newcommand{\TRrightlabel}[1]{%
445     % Place a strut in order to set line depth
446     \mbox{}\strut\settodepth{\T@peReleaseDepth}{\strut}%
447     \vadjust{\hspace{\textwidth}\hspace{\marginparsep}}%
448     \smash{\rlap{\TRwriteout{#1}}}}
449
450 % Internal command to typeset a TR flag in the left margin
451 \ifthenelse{\boolean{@twoside}}{%
452     % Two-sided document
453     \newcommand{\TRleftlabel}[1]{%
454         % Place a strut in order to set line depth
455         \mbox{}\strut\settodepth{\T@peReleaseDepth}{\strut}%
456         \vadjust{\smash{\llap{\TRwriteout{#1}}\kern\marginparsep}}}}
457 }%
458 % Otherwise, the command is the same as rightlabel
459 \newcommand{\TRleftlabel}[1]{%

```

```
460      \TRrightlabel{#1} }  
461 }
```

\TR Finally, the user-level command is simply \TR.

```
462 \ifthenelse{\boolean{RegisterTRFlags}}{  
463 \newcommand{\TR}[1]{%  
464 \stepcounter{T@peReleaseTag}%  
465 \label{TapeReleaseTag-\theT@peReleaseTag} %  
466 \ifthenelse{\isodd{\GetTRPageRef{TapeReleaseTag-\theT@peReleaseTag}}}{%  
467 {\TRrightlabel{TR\hspace{1pt}#1}}%  
468 {\TRleftlabel{TR\hspace{1pt}#1}}%  
469 } }  
470 {\newcommand{\TR}[1]{} }
```

Enjoy! Happy documenting.