Page Numbers and Other Pin Cites (`pages.dtx`)

Charles Duan

May 2, 2023

The often extraordinary length of legal documents, articles, and treatises necessitates citing specific portions of them in order to inform readers of the relevant parts of a reference that support the writer's claims. Yet the diversity of forms of legal references makes pin-citing the relevant parts a challenge. Some documents are paginated, others divided into sections, others marked with star-pagination reflecting archaic publications. And some forms of citation require identifying multiple pin cites for different published versions of a work—cases reported in several reporters, for example, or for session laws both the statutory section number and the page number of the *Statutes at Large*.

It is not desirable to put the burden on the user to manage formatting of page numbers and subdivisions. A legal citation system must understand users' page number inputs to compose tables of authorities, determine whether to omit page numbers in *id.* citations, and cite subsections of statutes, for example. Furthermore, there are many rules of pin-cite formatting that are inconvenient and burdensome for writers to remember and better suited for automatic processing—abbreviation of words like "paragraph" and italicization of "(*l*)," for example. As a result, a program such as this package must parse and interpret pin cites, and it thus must provide writers with a syntax for entering them.

This section describes the data model for pin cites, the input syntax that this system accepts for them, and the rules for formatting.

1 Overall Structure

Because of the diversity of types of legal document numbering systems, the data model and syntax for pin cites are fairly complex. For testing purposes, a writer may access the pin cite formatting algorithm using the command `\PageNumber{`⟨*number*⟩`}`.

\PageNumber

\@format@page@macro

The internal formatting command is `\@format@pageno`, described below. `\@format@page@macro{`⟨*command*⟩`}` is a shortcut when the page number is in a command sequence.

```
\def\PageNumber#1{\@format@pageno{#1}}
\def\@format@page@macro#1{\expandafter\@format@pageno\expandafter{#1}}%
```

A pin cite is made up of several "segments": a default segment and one or more named segments. The named segments are used for parallel citations; their

use is defined by specific reference types. For example, session laws use the default segment to hold the subsection number of the statute being cited, and then require a segment named `stat` that identifies the page of the *Statutes at Large* on which the subsection appears. The input syntax and an example are as follows:

⟨*pin-cite*⟩ := ⟨*segment*⟩ [`::`⟨*segment-name*⟩`:` ⟨*segment*⟩]*
Example: `S 12 ::stat: 783`

The example would cite § 12 of a statute, at page 783.

The overall objective of the pin cite formatting macros is to produce formatted page number text by pure expansion. That way, pin cites can be formatted inside `\edef` commands. Because these formatting macros date back to the very beginning of this citation program, before I had developed good TEX programming conventions and habits, these macros do not follow my usual practice of storing the formatted result and passing it back to a callback handler. Instead, they just drop the formatted text into the document directly.

\@format@pageno `\@format@pageno{`⟨*page*⟩`}` is the main entrypoint into the pin cite formatting algorithm. It extracts the default segment to be processed.

```
\def\@format@pageno#1{%
    \find@in{ ::}{#1}{%
        \@tworun\hi@fpg@onesegment\@gobble
    }{\hi@fpg@onesegment{#1}}%
}
```

\hi@fpg@segment `\hi@fpg@segment{`⟨*page*⟩`}{`⟨*segment-name*⟩`}{`⟨*prefix*⟩`}{`⟨*else*⟩`}` selects a named segment and formats it. If there is a named segment, it is prefixed with ⟨*prefix*⟩. If there is no such named segment, ⟨*else*⟩ is run.

```
\def\hi@fpg@segment#1#2#3#4{%
    \find@in{ ::#2: }{#1}{#3\@tworun\@gobble\@format@pageno}{#4}%
}
```

\hi@fpg@defsegment A reference type macro must call `\hi@fpg@defsegment` to declare that it will use a certain named segment.

```
\def\hi@fpg@defsegment#1{%
    \make@find@in{ ::#1: }%
}
\make@find@in{ ::}
```

A pin cite segment can consist either of a comma-separated list of items, or two comma-separated lists of items delimited by the word `to` (surrounded by spaces). The latter form is used for pin cites to ranges where each side of the range contains multiple items.

⟨*segment*⟩ := ⟨*list*⟩ [`to` ⟨*list*⟩]
⟨*list*⟩ := ⟨*item*⟩ [`,` ⟨*item*⟩]*
Example: `column 5, line 8 to column 6, line 5`

Here, the word “to” must be used to separate the ranges to avoid ambiguity.

```
\make@find@in{ to }
\def\hi@fpg@onesegment#1{%
    \find@in{ to }{#1}{\hi@fpg@to}{%
        \ifstrempty{#1}{}{\hi@fpg@split#1,\@stop{\hi@fpg@pagenumber{}{}}}%
    }%
}
\def\hi@fpg@to#1#2{%
    \@format@pageno{#1} to \@format@pageno{#2}%
}
```

Reads each pin cite item and runs `\hi@fpg@one` on each. Calls `\hi@fpg@emitstate` at the end of the list. #3 is the current page number state, described below.

```
\def\hi@fpg@split#1,#2\@stop#3{%
    %
    % In both of these branches of the conditional, we take advantage of the
    % fact that both |\hi@fpg@emitstate| and |\hi@fpg@split| receive the
    % state as the last argument, and \hi@fpg@one runs its callback with the
    % state as the last argument.
    %
    \ifblank{#1}{%
        % Ignore blank entries
        \ifblank{#2}{\hi@fpg@emitstate}{\hi@fpg@split#2\@stop}{#3}%
    }{%
        \hi@fpg@one#1\@stop{#3}{%
            \ifblank{#2}{\hi@fpg@emitstate}{\hi@fpg@split#2\@stop}%
        }%
    }%
    \hi@fpg@done
}
```

2 Pages and Divisions

Many forms of pin cite items are permitted, and how an item is interpreted may depend on what item preceded it. In general, there are two types of items in pin cites: page numbers and named divisions (like "table 5" or "§§ 15-20"). The number type is important for formatting: Ranges in page numbers are truncated, and named divisions need to be singular or plural and can accept subdivisions.

Generally, page numbers are written as bare numbers, and named divisions are written as the division name, a space, and one or more division numbers. A division number can follow a page number with or without a comma, or with an ampersand (indicating a citation to the page and the division). After a named division is used, any subsequent bare numbers will be interpreted as part of the division. To return to citing page numbers, insert the word `at` (e.g., `section 5, at 143`). This can be useful for helping readers find the location of text divisions. Thus:

Input	**Output**	**Explanation**
`157, 160-163`	157, 150–63	Page numbers
`line 403, 450-463`	ll. 403, 450–463	All are line numbers
`line 403, at 450-463`	l. 403, at 450–63	450–463 are pages
`157-158 table 1-2`	157–58 tbls.1–2	Citing just tables
`157-158 & table 1-2`	157–58 & tbls.1–2	Citing pages and tables

As seen in the above examples, the number part of either a page number or a named division can be a hyphenated range, and the hyphen is converted to a proper dash. To include an actual hyphen, use `\-` instead.

There are two problems to be solved when parsing pin cites: (1) figuring out whether a bare number is a page number or a division number, and (2) determining whether to emit a singular or plural version of a division name. To solve these problems, it is necessary to maintain some internal state wile formatting a pin cite. The state contains the following as grouped elements (other than the last element):

1. A macro to be run if a bare number is found, to format the number. The macro receives two arguments: the number text, and the loop continuation callback.

2. Reserved text to be emitted when a new division name is encountered. Typically this is used to store the singular form of a division name (which cannot be emitted immediately since a subsequent bare number would require a plural form).
3. Reserved text to be emitted if a bare number is found that proves a previously encountered division name to require a plural.
4. Trailing matter to be emitted when any of the reserved text is emitted.

The basic idea is that if a bare number is encountered, items 3 and 4 from the state are emitted (since the bare number proves that any prior division should be pluralized) and macro 1 should be run. If a new division name is encountered, however, items 2 and 4 are emitted before processing the division name and number.

This macro is used whenever the non-plural reserved text should be emitted from the state.

```
\def\hi@fpg@emitstate#1{%
    \@secondofthree#1%
}
```

Processes a single pin cite item. #1 and #2 are the pin cite item number, split into two arguments both to consume any leading spaces and to identify the input type: a bare number if #1 is a digit and a named division if #2 is a letter. (Special handling is given to punctuation, as described below.) #3 is the internal state that is received. #4 is a callback that must be run at the end of processing, to continue the comma splitting function.

```
\def\hi@fpg@one#1#2\@stop#3#4{%
    \find@start{at }{#1#2}{\hi@fpg@at}{%
        \@ifletter{#1}{\hi@fpg@letter{#1#2}}{%
            \@ifdigit{#1}{\hi@fpg@number{#1#2}}{%
                \hi@fpg@symbol{#1}{#2}%
            }%
        }%
    }{#3}{#4}% Arguments to all macros
}
\make@find@start{at }
\make@find@in{ }
```

If the pin cite item starts with “at,” then this is run to flush the state and display the number as a page number. #1 is the page number, #2 the state, and #3 the callback.

```
\def\hi@fpg@at#1#2#3{%
    \hi@fpg@emitstate{#2}%
    at % space
    \hi@fpg@pagenumber{#1}{#3}%
}
```

This is run upon reading a bare number in a citation item. It expands the state, emits the plural form, and runs the macro in the state to format the number. #1 is the number, #2 is the state, and #3 is the callback.

```
\def\hi@fpg@number#1#2#3{%
    \hi@fpg@number@#2\@stop{#1}{#3}%
}%
```

#1 and #2 are read from the state contents, so #1 is the macro to run and #2 are the two state elements and any trailing text. #3 is the text of the page number, and #4 the callback.

```
\def\hi@fpg@number@#1#2\@stop#3#4{%
    \@secondoftwo#2%
    #1{#3}{#4}%
}
```

This is called when the first character of a pin cite item is a letter (or otherwise indicates a division name), such as “tbl. 1”. Generally it emits reserved text in the state and then processes a division. However, if the “division” name is the word `at`, then the number is treated as a page number rather than a named division. #1 is the division name and number, #2 the internal state, and #3 the callback to continue processing the page number.

```
\def\hi@fpg@letter#1#2#3{%
    \hi@fpg@emitstate{#2}%
    \hi@fpg@division{#1}{\hi@fpg@nextdivision}{#3}%
}
```

Further fine-tuning of the pin cite parsing algorithm can be done by placing special codes or punctuation at the beginning of pin cite items:

Input	Type	Explanation
`S 3`	Division	Produces § 3
`P 3`	Division	Produces ¶ 3
`-34/a`	Division	Interpreted as a division with no name: 34(a)
`!A5`	Number	Uses text after the exclamation point as a bare number, and does not format at all. Useful for newspaper articles
`*16`	Number	Treats whatever follows the star as a page number. Useful for star-paginated works
`?`	Missing	Issues a warning to remind the writer to fill in a pin cite
`/a3`	Division	Interpreted as a subdivision: (a)(3). Must follow a named division

Handles punctuation at the start of a pin cite item. #1 is the punctuation, #2 the rest of the page number, #3 the state, #4 the callback.

```
\def\hi@fpg@symbol#1#2#3#4{%
    \find@eq{/}{#1}{\hi@fpg@subsection{#2}{#3}{#4}}{%
        \find@try\find@eq{%
            {-}{\hi@fpg@letter{ #2}{#3}{#4}}%
            {!}{\hi@fpg@number{\hi@pgnum@raw#2}{#3}{#4}}%
            {*}{\hi@fpg@emitstate{#3}*\hi@fpg@pagenumber{#2}{#4}}%
            {?}{%
                \hi@pgnum@raw % so atorsect works
                \protect\hi@fpg@missingpg{#1#2}\hi@fpg@plain{#1#2}{#3}{#4}%
            }%
        }{#1}{%
            % None of the tests succeeded
            \protect\hi@fpg@unexpectedchar{#1#2}%
            \hi@fpg@plain{#1#2}{#3}{#4}%
        }%
    }%
}
\make@find@eq{-}
\make@find@eq{/}
\make@find@eq{*}
\make@find@eq{!}
\make@find@eq{?}
```

This is just a marker used to identify page numbers that don't look like page numbers. It is used by `\hi@atorsect` and similar commands.

```
\def\hi@pgnum@raw{\noexpand\hi@pgnum@raw}
```

These produce warnings when parsing pin cite items.

```
\def\hi@fpg@unexpectedchar#1{%
    \PackageWarning\hi@pkgname{%
        Page number `#1' starts with an unexpected character.\MessageBreak
        If it is meant to be a page number, then precede it\MessageBreak
        with the `!' symbol. It appeared%
    }%
}
\def\hi@fpg@missingpg#1{%
    \@expandarg\hi@missing@page{\@this@case}%
}
```

3 Subdivisions

Named divisions of text are sometimes organized into top-level divisions and subdivisions. For example, "§ 5(a)" refers to subsection (a) of the fifth section of a reference. Distinguishing between top-level divisions and subdivisions is necessary because it affects pluralization of the division name ("§§ 5–6," but

“§ 5(a)–(b)”). Additionally, writing all the parentheses in conventional subdivision numbers is somewhat cumbersome.

For input to this package, subdivisions are separated from top-level divisions with a slash. Bare subdivisions (without a top-level subdivision) may be used as pin cite items to cite multiple non-contiguous subdivisions, and ranges may include bare subdivisions. Single characters in a subdivision will be automatically parenthesized. The following are examples:

Input	Output
`S 5/a`	§ 5(a)
`S 5/a-6/b`	§§ 5(a)–6(b)
`S 5/a-/b`	§ 5(a)–(b)
`S 5/a, /c`	§ 5(a), (c)
`S 5/a1A`	§ 5(a)(1)(A)
`S 5/a1A(iii)`	§ 5(a)(1)(A)(iii)
`S 5/a1A{note}`	§ 5(a)(1)(A)note

4 Formatting

We begin with some utility macros. `\hi@fpg@done` marks the end of the pin cite formatting. `\hi@fpg@comma` is used to insert a comma into a pin cite; the comma is suppressed if it detects `\hi@fpg@done` is upcoming.

```
\def\hi@fpg@done{\iffalse\hi@fpg@done\fi}
\def\hi@fpg@comma#1{\ifx\hi@fpg@done#1\else, \fi#1}
```

Outputs a single number to the current text. This performs replacements for special sequences in page numbers, such as (which becomes -).

```
\def\hi@fpg@singlenum#1{%
    \find@in@cs{/-}{#1}{\hi@fpg@singlenum@dash}{#1}%
}
\def\hi@fpg@singlenum@dash#1#2{%
    #1-\hi@fpg@singlenum{#2}%
}
```

Emits unformatted text as part of the pin cite, handling state accordingly (by emitting any reserved text and anticipating a page number as the next input). #1 is the page number, #2 is the state, #3 the callback.

```
\def\hi@fpg@plain#1#2#3{%
    \hi@fpg@emitstate{#2}%
    #1% Output the page number as specified
    #3{\hi@fpg@pagenumber{}{}\hi@fpg@comma}%
}
```

4.1 Page Numbers

Formats an ordinary page number. This performs three searches to determine the type of ordinary page number:

- Ampersand (123 & n.456)
- Space (123 note 456)
- Otherwise the whole text is a single page number

#1 is the number, and #2 is the callback.

```
\def\hi@fpg@pagenumber#1#2{%
    \find@in{ & }{#1}{%
        \hi@fpg@pgno@withdivision{ \& }%
    }{%
        \find@in{ }{#1}{%
            \hi@fpg@pgno@withdivision{ }%
        }{\hi@fpg@pgno@nosuffix{#1}}%
    }{#2}%
}
\make@find@in{ }
\make@find@in{ & }
```

4.1(a) Parsing Page Suffixes If there is an ampersand or space, then we output the page number, output the ampersand or space, and process the remainder with the expectation that it is a letter-starting page reference (e.g., 123 n.5). #1 is the divider (ampersand or space), #2 the pre-ampersand page number, #3 the post-ampersand division, and #4 the callback.

```
\def\hi@fpg@pgno@withdivision#1#2#3#4{%
    \chop@space@then@run{#2}\hi@fpg@pgrange
    #1%
    \hi@fpg@division{#3}{\hi@fpg@nextdivisionamp}{#4}%
}
```

No division following the number. Format the page number and execute the callback, with state updated.

```
\def\hi@fpg@pgno@nosuffix#1#2{%
    \chop@space@then@run{#1}\hi@fpg@pgrange
    #2{\hi@fpg@pagenumber{}{}\hi@fpg@comma}%
}
```

4.1(b) Page Ranges These macros deal with the truncation of page numbers from the ends of ranges and other formatting of page ranges.

Searches for a range hyphen. If found, formats the first part and then passes the second part for truncation. Otherwise, formats just the single number.

```
\def\hi@fpg@pgrange#1{%
    \find@in{-}{#1}\hi@fpg@pgrange@withdash{\hi@fpg@singlenum{#1}}%
}
\make@find@in{-}
\def\hi@fpg@pgrange@withdash#1#2{%
    \hi@fpg@singlenum{#1}%
    \find@in@cs{/-}{#1#2}{ to \@gobbletwo}{--}%
    \hi@fpg@toend#1\@mark\@mark#2\@mark\@mark\@stop
}
```

Computes the last number of a page range: It turns 12345-12346 into 46 and 12345-12456 into 456. The following are the arguments to the macro:

- #1#2 is the remaining part of the first number.
- #3 is the part of the second number that is different from the first number.
- #4#5 is the remaining part of the second number.
- #6 is the part of the second number that is the same as the first number.
- #7 is the part of the second number that is the same as the first number, except we'll keep it in case the similar part's too short.

The rules are:

1. If we reach the end of the first number (#2 is empty),
    (a) and we reach the end of the second number at the same time (#5 is empty), then we just print #3#4, the differing part of the second number.
    (b) and we haven't reached the end of the second number, then they're different lengths, and we print the whole of the second number (i.e., #6#3#4#5).

2. Otherwise, if we reach the end of the second number, then they're also different lengths and we print all of the second number.
3. Otherwise we haven't reached any of the ends of numbers.
    (a) If there is anything in the dissimilar part of the second number, then skip the next sentence.
    (b) Otherwise, if the first digit of the first number (#1) is equal to the first digit of the second number (#4), then push #4 onto #6.
    (c) Otherwise, push #4 onto #3.
    (d) Now discard #1, and try again.

```
\def\hi@fpg@toend#1#2\@mark#3\@mark#4#5\@mark#6\@mark#7\@stop{%
    \ifstrempty{#2}{% If we're done traversing the first number
        \ifstrempty{#5}{% If we're done traversing the second number
            \ifstrempty{#3}{#7}{}% If #3 is empty, throw in #7
            \hi@fpg@singlenum{#3#4}%
        }{\hi@fpg@singlenum{#6#7#3#4#5}}% Print the numbers in full
    }{% We're not done traversing the first number
        % If we're done traversing the second number, print them in full
        \ifstrempty{#5}{\hi@fpg@singlenum{#6#7#3#4#5}}{%
            % We're not done with either number
            \ifstrempty{#3}{% If there's no dissimilar stuff
                \ifx#1#4% If the first digits are the same
                    \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi
                {\hi@fpg@toend#2\@mark\@mark#5\@mark#6#7\@mark#4\@stop}%
                    % First two digits are different
                {\hi@fpg@toend#2\@mark#4\@mark#5\@mark#6#7\@mark\@stop}%
            }{% There is some dissimilar stuff
                \hi@fpg@toend#2\@mark#3#4\@mark#5\@mark#6#7\@mark\@stop
            }%
        }%
    }%
}
```

5 Divisions

Draws a numbered division with names. If no division number is given, just emits the division name as abbreviated, and assumes that any subsequent bare numbers are page numbers. If a division number is given, then we have to do special comma handling since the next-state macro might prefer placing ampersands. As a result, commas are not included in the trailing state.

#1 is the division name and number, #2 is the macro for the next state, and #3 is the callback.

```
\def\hi@fpg@division#1#2#3{%
    \find@in{ }{#1}{%
        \hi@fpg@division@numbered{#2}{#3}%
    }{%
        \hi@fpg@divname{#1}{}%
        #3{\hi@fpg@pagenumber{}{}\hi@fpg@comma}%
    }%
}
%    \begin{macrocode}
%
% \#1 is the next state macro, \#2 the callback, \#3 the division name, \#4 the
% division number.
%
%    \begin{macrocode}
\def\hi@fpg@division@numbered#1#2#3#4{%
    \hi@fpg@ifrange{#4}{%
        %
        % For a range, output the plural division name (it must be plural) and
        % the division number, and proceed to the next chunk. No comma is put
        % into the plural form state because we don't know if it should be an
        % ampersand.
        %
        \hi@fpg@divname{#3}\hi@fpg@plural
        \hi@fpg@divno{#4}%
        #2{#1{\hi@fpg@comma}{}}%
    }{%
        % If it's not a range, then we don't know if it will be singular or
        % plural. Update the state to include the singular and plural values and
        % then proceed to the next chunk.
        #2{#1{%
                \hi@fpg@divname{#3}\hi@fpg@singular
                \hi@fpg@divno{#4}\hi@fpg@comma
```

```
            }{%
                \hi@fpg@divname{#3}\hi@fpg@plural
                \hi@fpg@divno{#4}%
            }%
        }%
    }%
}
\make@find@in{-}
\make@find@in{-/}
```

Determines if the given section number specification (#1) is a range. It is a range if (1) it includes a dash, and (2) the matter immediately trailing the dash is not a slash.

```
\def\hi@fpg@ifrange#1#2#3{%
    \find@in{-/}{#1}{\@firstofthree{#3}}{%
        \find@in{-}{#1}{\@firstofthree{#2}}{#3}%
    }%
}
```

Formats a division name. #1 is the name, and #2 a macro indicating whether to choose singular or plural (or empty to choose the default form).

```
\def\hi@fpg@divname#1#2{%
    \@ifundefined{hi@div@#1}{%
        \ifstrempty{#1}{}{%
            \protect\hi@fpg@warngraph{#1}%
            #1\space
        }%
    }{%
        \ifstrempty{#2}{%
            \@expand\chop@space{\csname hi@div@#1\endcsname}{iii}%
        }{%
            \@expand{\hi@fpg@handlespc#2}{\csname hi@div@#1\endcsname}{ii}%
        }%
    }%
}
\def\hi@fpg@handlespc#1#2{%
    \@expand\hi@fpg@handlespc@{#1{#2}}{ii}%
}
\def\hi@fpg@handlespc@#1{%
    \find@end{ }{#1}{\@unbrace}{#1\@gobble}{~}%
}
\def\hi@fpg@plural#1{\@thirdofthree#1}
\def\hi@fpg@singular#1{\@secondofthree#1}
\def\hi@fpg@warngraph#1{%
    \PackageWarning\hi@pkgname{%
        Word `#1' is not a defined division name.\MessageBreak
        Consider placing an exclamation mark before the\MessageBreak
        page number to avoid this message.\MessageBreak
        It appeared}%
}
```

Processes a subsequent bare number after a named division. Note that no comma was drawn from the previous state, so we need to add one and also not include one in the plural-form element of the state. #1 is the number to be formatted, and #2 the callback.

```
\def\hi@fpg@nextdivision#1#2{%
    , \hi@fpg@divno{#1}%
    #2{\hi@fpg@nextdivision{\hi@fpg@comma}{}}%
}
```

Processes a subsequent bare number after a named division, where an ampersand is desired before the last element. Rather than emitting the division, we save the text to be emitted to the state. #1 is the number to be formatted, and #2 the callback.

```
\def\hi@fpg@nextdivisionamp#1#2{%
    #2{\hi@fpg@nextdivisionamp{%
            \space\&\space
            \hi@fpg@divno{#1}\hi@fpg@comma
        }{%
            , \hi@fpg@divno{#1}%
        }%
    }%
}
```

Formats a division number. First, determine if there's a dash. If so, then split it and format as if two separate division numbers, separated by an en-dash or the word "to" as appropriate. Otherwise, format as a single subdivision number.

```
\def\hi@fpg@divno#1{%
```

```
    \find@in{-}{#1}{\hi@fpg@divno@dash}{\hi@fpg@divno@single{#1}}%
}
\make@find@in{-}
\def\hi@fpg@divno@dash#1#2{%
    \hi@fpg@divno@single{#1}%
    \find@in@cs{/-}{#1#2}{ to \@gobbletwo}{--}%
    \hi@fpg@divno@single{#2}%
}
\make@find@in@cs{/-}{\-}
```

Formats a single (non-range) subdivision number. Determine if there is a slash. If so, then output the pre-slash content raw, plus the parenthesized post-slash content. Otherwise just output the content raw.

```
\def\hi@fpg@divno@single#1{%
    \find@in{/}{#1}{%
        \@tworun\hi@fpg@singlenum\hi@parenize
    }{\hi@fpg@singlenum{#1}}%
}
```

5.1 Subdivisions

Formats for a bare subdivision. It is assumed that the full division name was previously cited. The problem is that citation of a subdivision alone does not indicate to us whether the plural or singular form of the full division name is needed. Thus, rather than actually producing any output, this macro just augments the state to add the subdivision to both the plural and singular forms.

#1 is the subsection (without the slash); #2 is the state; #3 is the callback.

```
\def\hi@fpg@subsection#1#2#3{%
    \hi@fpg@subsection@alterstate{#1}{#3}#2\@stop
}
```

#1 is the subdivision, #2 is the callback, #3-#5 the state elements, and #6 any trailing material from the state.

```
\def\hi@fpg@subsection@alterstate#1#2#3#4#5#6\@stop{%
    #2{%
        {#3}%
        {#4\hi@parenize{#1}\hi@fpg@comma}%
        {#5\hi@fpg@comma\hi@parenize{#1}}%
        #6%
    }%
}
```

Traverses a string and puts parentheses around its elements. So A12e(i)n becomes (A)(12)(e)(i)(n).

```
\def\hi@parenize#1{%
    \ifstrempty{#1}{}{%
        \hi@psize#1\@stop s%
    }%
}
```

Processes parenthesization for each character of a string. #1 is the character being processed, #2 is the rest of the string, and #3 is a state variable of “s”, “o”, or “n” indicating whether this is the start, outside a parenthesis group, or inside a parenthesis group (i.e. inside a number).

```
\def\hi@psize#1#2\@stop#3{%
    \@ifonechar{#1}{%
        \@testcase
        \ifx#1(\fi {% )
            \hi@psize@close#3\hi@psize@paren#2\@stop
        }%
        \ifx#1[\fi {% ]
            \hi@psize@close#3\hi@psize@bracket#2\@stop
        }%
        \default{%
            \@ifletter{#1}{%
                \hi@psize@close#3(\hi@psize@ltr{#1})%
                \ifstrempty{#2}{}{\hi@psize#2\@stop o}%
            }{%
                \hi@psize@noopen#3#1%
                \ifstrempty{#2}{)}{\hi@psize#2\@stop n}%
            }%
        }%
    }{%
        \hi@psize@close#3#1\ifstrempty{#2}{}{\hi@psize#2\@stop o}%
    }%
}
```

When inside an explicit parenthesis group, skips to the closing parenthesis and renders the text.

```
\def\hi@psize@paren#1)#2\@stop{(#1)\ifstrempty{#2}{}{\hi@psize#2\@stop o}}
\def\hi@psize@bracket#1]#2\@stop{[#1]\ifstrempty{#2}{}{\hi@psize#2\@stop o}}
```

Closes the current group if inside one (i.e., the state variable #1 is “n”).

Also adds a penalty allowing a break if not at the start (i.e., state variable is not “s”). This works here because `\hi@psize@close` is used immediately before opening any parenthesis group other than a number group.

```
\mathchardef\hi@psize@penaltysize=9999
\def\hi@psize@penalty{\penalty\hi@psize@penaltysize}
\def\hi@psize@close#1{\ifx#1n)\fi\ifx#1s\else\hi@psize@penalty\fi}
```

Opens a new parenthesis group unless already inside one (state variable is “n”). Also add a penalty allowing a break if not at the start.

```
\def\hi@psize@noopen#1{\ifx#1n\else\ifx#1s\else\hi@psize@penalty\fi(\fi}
```

Italicizes the letter ”l”.

```
\def\hi@psize@ltr#1{\ifx#1l\hi@fn@ell{l}\else#1\fi}
```

5.2 Abbreviations

Set up table of division names.

A macro is defined for all forms of each abbreviation. Each macro has three parts: a selector, the singular form, and the plural form. If the macro is simply executed, the selector will choose the singular or plural form based on the macro that was called. However, the specific forms may be acquired using `\hi@fpg@singular` or `\hi@fpg@plural`.

```
\def\hi@abbrev#1#2#3#4{%
    \@namedef{hi@div@#1}{\@firstoftwo{#2}{#4}}%
    \@namedef{hi@div@#3}{\@secondoftwo{#2}{#4}}%
    \csletcs{hi@div@\chop@space{#4}}{hi@div@#3}%
    \csletcs{hi@div@\chop@space{#2}}{hi@div@#1}%
}
\input hi-divisions
%
% Special definitions for S and P
%
\def\reserved@a#1#2{%
    \@namedef{hi@div@#1}{\@firstoftwo{#2 }{#2#2 }}%
    \@namedef{hi@div@#1#1}{\@secondoftwo{#2 }{#2#2 }}%
}
\reserved@a S\textsection
\reserved@a P\textparagraph
```

1 Pin Cite Subdivision Joining

For `alias` and some statute citations, it is possible to give an overall division or page number for the reference and then refer to a specific part of it. For example, one might define the statute 35 U.S.C. § 112 as a reference `sec-112`, and then wish to refer to subsection (b) of that statute. To do so, one may write:

> `\sentence{sec-112 at /b}`
> Equivalent to: `\sentence{sec-112 at S 112/b}`
> Produces: 35 U.S.C. § 112(b)

In other words, the pin cite given as part of the citation item is joined to the pin cite given in the reference definition. The following rules define how a reference pin cite is joined with a citation item pin cite:

- If the citation item pin cite does not start with a slash indicating a subdivision, then the reference pin cite is discarded and only the citation item’s is used.

- If the citation item pin cite starts with a slash and the reference pin cite has no slash, then the two are concatenated.
- If the citation item pin cite starts with a slash and the reference pin cite has a slash, then the slash is removed from the citation item pin cite and the two are concatenated.

Thus, the following examples:

Reference	Citation Item	Result
`S 112`	`S 102/a`	§ 102(a)
`S 112`	`/b`	§ 112(b)
`S 505/j`	`/2A`	§ 505(j)(2)(A)
`S 505/j`	`/2A-/3`	§ 505(j)(2)(A)–(3)

The range joins correctly not because of these rules but because of the ordinary parsing rules for subdivision ranges, given above.

Note that the joining algorithm performs no error checking, and will join subdivisions to page numbers even though subdivisions make no sense in that context. Additionally, this feature is only supported by certain reference types that enable it.

```
\def\hi@pages@join#1#2{%
    \find@start{/}{#2}{%
        \find@in{/}{#1}%
            {\hi@pages@join@bothslash}%
            {\def\@this@orig@page{#1#2}\@gobble}%
    }{%
        \ifstrempty{#2}{\def\@this@orig@page{#1}}{\def\@this@orig@page{#2}}%
    }%
    \protected@edef\@this@page{\@format@page@macro\@this@orig@page}%
}
\make@find@start{/}
\make@find@in{/}
\def\hi@pages@join@bothslash#1#2#3{%
    \def\@this@orig@page{#1/#2#3}%
}
\def\TestJoinDivisions#1#2{%
    \begingroup
        \hi@pages@join{#1}{#2}%
        \@this@page
    \endgroup
}
```

1 Formatting Around Pin Cites

Pin cites can affect the text that precedes the pin cite number. In many situations, standard page numbers are often preceded by the word “at” whereas named subdivisions are not. The macro `\hi@atorsect{`⟨*pincite*⟩`}` is provided to insert “at” depending on the nature of ⟨*pincite*⟩, and `\hi@page@atorsect` performs that function on the currently active pin cite in a citation item.

`\hi@atorsect`

`\hi@page@atorsect`

Write the word “at” unless `\@this@page` starts with a paragraph or section mark (or other section indicator).

```
\DeclareRobustCommand\hi@page@atorsect{%
    \expandafter\hi@atorsect@\@this@page.\@stop
}
\def\hi@atorsect#1{\hi@atorsect@#1.\@stop}
\def\hi@atorsect@withpage#1{\hi@atorsect{#1}#1}%
```

```
\make@find@eq{*}
\make@find@eq{?}
\def\hi@atorsect@#1#2\@stop{%
    \@ifdigit{#1}\@iden{%
        \find@try\find@eq{{*}\@iden{?}\@iden}{#1}{%
            \@test\ifx\hi@pgnum@raw#1\fi{\@iden}{\@gobble}%
        }%
    }{at }%
}
```

For other citation types such as books, the pin cite is separated from text in the citation with just a space. However, there are exceptions:

- If the preceding text ends with a digit, then a comma is placed, and "at" is placed if the pin cite is a page number as described above.
- If the preceding text is "R." (for "Record"), then "at" is placed regardless of the type of pin cite.[1]

Thus, the following examples:

Preceding Text	**Pin Cite**	**Output**
`Book`	`15`	Book 15
`Treatise`	`S 5`	Treatise § 5
`Annual Report 2015`	`35`	Annual Report 2015, at 35
`Agency Report 2006`	`S 12`	Agency Report 2006, § 12
`R.`	`102`	R. at 102
`R.`	`paragraph 36`	R. at para. 36

This macro is to be used in a citation macro definition, and must occur inside `\hi@maybepage` so it can be assumed that a page number has been given. #1 is a macro to be expanded containing the preceding text to be tested. It depends on `\hi@page@atorsect` being robust so that it remains unexpanded while the citation macro is being defined.

```
\def\hi@page@space#1{%
    \expandafter\hi@page@space@\expandafter{#1}%
}
\def\hi@page@space@#1{%
    \@ifendswithdigit{#1}{, \hi@page@atorsect}{%
        \find@eq{R.}{#1}{ at }{\space}%
    }%
}
\make@find@eq{R.}
%
% Tests whether the argument ends with a digit.
\def\@ifendswithdigit#1{%
    \find@try\find@end{%
        0\@secondofthree 1\@secondofthree 2\@secondofthree 3\@secondofthree
        4\@secondofthree 5\@secondofthree 6\@secondofthree 7\@secondofthree
        8\@secondofthree 9\@secondofthree
    }{#1}\@secondoftwo
}
\make@find@end{0} \make@find@end{1} \make@find@end{2} \make@find@end{3}
\make@find@end{4} \make@find@end{5} \make@find@end{6} \make@find@end{7}
\make@find@end{8} \make@find@end{9}
```

Finally, it is sometimes useful to expand initial `\textsection` and `\textparagraph` symbols to words. This is useful for statute citations. The command `\hi@expand@symbols{`⟨*pincite*⟩`}{`⟨*callback*⟩`}` will do this.

`\hi@expand@symbols`

```
\DeclareRobustCommand\hi@expand@symbols{\hi@expand@symbols@}
```

[1] For Record cites to named subdivisions (e.g., "R. at para. 5"), it is unclear whether "at" should be included. It seems preferable to do so to ensure that "R." is not interpreted as "Rule."

```
\def\hi@expand@symbols@#1#2{%
    \find@try\find@start@cs{%
        {SS}{\hi@expand@symbols@@{#2}{sections}{SS}}%
        {PP}{\hi@expand@symbols@@{#2}{paragraphs}{PP}}%
        {S}{\hi@expand@symbols@@{#2}{section}{S}}%
        {P}{\hi@expand@symbols@@{#2}{paragraph}{P}}%
    }{#1}{#2{#1}}%
}
\def\hi@expand@symbols@@#1#2#3#4{%
    #1{#2#4}%
}
\make@find@start@cs{S}{\textsection}
\make@find@start@cs{SS}{\textsection\textsection}
\make@find@start@cs{P}{\textparagraph}
\make@find@start@cs{PP}{\textparagraph\textparagraph}
```

The remaining items are lower level macros for conditional citation outputs based on the content of pin cites.

`\hi@maybepage` Shows ⟨*text*⟩`\@this@page` only if `\@this@page` is not `\relax`. Also, if `\@this@page` ends with a dot, sets `\@hi@dottrue`.

```
\DeclareRobustCommand\hi@maybepage[1]{%
    \@test\ifx\@this@page\relax\fi{}{%
        \ifstrempty{#1}{\expandafter\@capnext}{#1\hi@dot@set{#1}}\@this@page
        \hi@dot@set@page
    }%
}
```

Performs the first argument if `\@this@page` is set, and the second argument otherwise.

```
\DeclareRobustCommand\hi@ifpage{%
    \ifx\@this@page\relax
        \expandafter\@secondoftwo \else
        \expandafter\@firstoftwo\fi
}
```

Shows either ⟨*#1*⟩`\@this@page` or ⟨*#2*⟩, depending on whether `\@this@page` is set.

```
\DeclareRobustCommand\hi@pageordefault[2]{%
    \@test\ifx\@this@page\relax\fi{%
        \ifstrempty{#2}{}{\@capnext#2\hi@dot@set{#2}}%
    }{%
        \ifstrempty{#1}{\expandafter\@capnext}{#1\hi@dot@set{#1}}\@this@page
        \hi@dot@set@page
    }%
}
```

Like `\hi@pageordefault`, but will expand `\@this@page` (only; won't expand #2) for section and paragraph, placing a `\@capnext` in front.

```
\DeclareRobustCommand\hi@expandedpageordefault[2]{%
    \@test\ifx\@this@page\relax\fi{%
        \ifstrempty{#2}{}{\@capnext#2\hi@dot@set{#2}}%
    }{%
        \ifstrempty{#1}{%
            \expandafter\hi@expand@symbols\expandafter{\@this@page}{%
                \expandafter\@capnext\@iden
            }%
        }{%
            #1\expandafter\hi@expand@symbols\expandafter{\@this@page}\@iden
        }%
        \hi@dot@set@page
    }%
}
```