The bibleref-mouth package *

Chad Parry
spam@chad.parry.org

February 26, 2012

Abstract

The bibleref-mouth package is a LaTeX 2ε library that allows Bible references to be formatted in a consistent way. It is similar to the bibleref package, except that the formatting macros are all purely expandable. That is, they are all implemented in TeX's mouth. This means that they can be used in any expandable context, such as an argument to the `\url` command. To write a reference, the document should use the command `\bibleref{`⟨*reference*⟩`}`.

1 Introduction

Two excellent packages already exist for formatting Bible references: bibleref by Nicola Talbot and bibleref-parse by Sebastian Kuhnert. This package draws heavily from the ideas in both of those. Those packages should meet any reasonable requirement for formatting a Bible reference, except for one: they do not work in an expandable context. This limitation occasionally turns into a problem, for example in these situations:

- A Bible reference gets written within an `\edef` command
- A Bible reference is part of a section heading
- A Bible reference is used within a hypertext reference

The last situation may sound contrived, but it is reasonable that an informal document might wish to link to an online Bible, such as at Bible Gateway. This package supports such a linked reference. For example, a hypertext reference like the following can be produced by the command below: Genesis 1:27.

```
\setbiblestyle{biblegateway.com}
\bibleref{Gen 1:27}
```

2 Usage

For a reasonable default experience, it is sufficient to know only one command.

*This document corresponds to bibleref-mouth v1.0, dated 2012/02/26.

`\bibleref` The document should create a Bible reference by calling `\bibleref{`⟨*reference*⟩`}`. The entire reference should be passed as a single argument, even if it contains a range or list of verses.

Books can be spelled out or abbreviated. A list of supported abbreviations is given in Table 1. Numbered books can be specified with Arabic or Roman numerals, with an optional space separating the number from the book. Abbreviations may have an optional trailing period. For example, it is equally acceptable to say `IISamuel`, `IISam.`, `IISam`, `2Samuel`, `2Sam.`, `2Sam`, `II Samuel`, `II Sam.`, `II Sam`, `2 Samuel`, `2 Sam.`, or `2 Sam`. Regardless of the format used in the `\bibleref` command, the reference will be formatted according to the active reference style. New custom books and abbreviations can be added, according to the pattern shown in section 3.8.1.

References must contain a book name. They may optionally contain a chapter, and if they contain a chapter then they may optionally contain a verse. References may be created for a range from one passage to another. References may also be created for a list of passages. A space should separate the book name from the chapter number. (Spaces will actually be ignored by the parser, so it is also allowed to write the book and the chapter with no separation between them). The chapter and verse should be separated by a `:` character, and optionally spaces. The begin and end passages of a range should be separated by a `-` character, and optionally spaces. Two passages that are part of a list should be separated by a `;` character, and optionally spaces. If the passages in a range or a list are from the same book, then the book does not need to be specified for the second passage. Likewise, if the passages in a range or a list have the same book and chapter, then the chapter does not need to be specified for the second passage. However, if two passages are in a list and they have the same book and chapter, then they should be separated by a `,` character instead of a `;` character. This helps resolve the ambiguity around whether the number following the delimiter represents a chapter or a verse.

The following examples show most of the different reference types that are supported by the parser.

- Single book: `\bibleref{Genesis}`
- Single chapter: `\bibleref{Exodus 32}`
- Single verse: `\bibleref{Amos 3:7}`
- Multiple books: `\bibleref{Matt; Mark; Luke; John}`
- Multiple chapters: `\bibleref{Psalms 2; 16; 32; 69; 95; 110}`
- Multiple verses within a chapter: `\bibleref{Matthew 5:13,48}`
- Multiple verses across chapters: `\bibleref{John 7:17; 14:15; 17:3}`
- Multiple different references: `\bibleref{Isa. 9:6; Ezek. 37; Rev.}`
- Range of books: `\bibleref{Genesis - Deuteronomy}`

Table 1: Supported Abbreviations

Book	**Long Abbreviation**	**Short Abbreviation**
Genesis	Gen.	Gn.
Exodus	Exod.	Ex.
Leviticus	Lev.	Lv.
Numbers	Num.	Nb.
Deuteronomy	Deut.	Dt.
Joshua	Josh.	Jos.
Judges	Judg.	Jg.
Ruth	Ruth	Rt.
1 Samuel	1 Sam.	1 S.
2 Samuel	2 Sam.	2 S.
1 Kings	1 Kgs.	1 K.
2 Kings	2 Kgs.	2 K.
1 Chronicles	1 Chr.	1 Ch.
2 Chronicles	2 Chr.	2 Ch.
Ezra	Ezra	Ezr.
Nehemiah	Neh.	Ne.
Tobit	Tobit	Tb.
Judith	Judith	Jdt.
Esther	Esther	Est.
1 Maccabees	1 M.	1 M.
2 Maccabees	2 M.	2 M.
Job	Job	Jb.
Psalms	Ps.	Ps.
Proverbs	Prov.	Pr.
Ecclesiastes	Eccles.	Qo.
SongofSongs	S. of S.	Sg.
Wisdom	Wisd.	Ws.
Ecclesiasticus	Ecclus.	Si.
Isaiah	Isa.	Is.
Jeremiah	Jer.	Jr.
Lamentations	Lam.	Lm.
Baruch	Baruch	Ba.
Ezekiel	Ezek.	Ezk.
Daniel	Dan.	Dn.
Hosea	Hos.	Ho.
Joel	Joel	Jl.
Amos	Amos	Am.
Obadiah	Obad.	Ob.
Jonah	Jonah	Jon.
Micah	Mic.	Mi.
Nahum	Nah.	Na.
Habakkuk	Hab.	Hab.
Zephaniah	Zeph.	Zp.

Table 1: Supported Abbreviations (continued)

Book	**Long Abbreviation**	**Short Abbreviation**
Haggai	Hag.	Hg.
Zechariah	Zech.	Zc.
Malachi	Mal.	Ml.
Matthew	Matt.	Mt.
Mark	Mark	Mk.
Luke	Luke	Lk.
John	John	Jn.
Acts	Acts	Ac.
Romans	Rom.	Rm.
1 Corinthians	1 Cor.	1 Co.
2 Corinthians	2 Cor.	2 Co.
Galatians	Gal.	Ga.
Ephesians	Eph.	Ep.
Philippians	Phil.	Ph.
Colossians	Col.	Col.
1 Thessalonians	1 Thess.	1 Th.
2 Thessalonians	2 Thess.	2 Th.
1 Timothy	1 Tim.	1 Tm.
2 Timothy	2 Tim.	2 Tm.
Titus	Tit.	Tt.
Philemon	Philem.	Phm.
Hebrews	Heb.	Heb.
James	Jas.	Jm.
1 Peter	1 Pet.	1 P.
2 Peter	2 Pet.	2 P.
1 John	1 John	1 Jn.
2 John	2 John	2 Jn.
3 John	3 John	3 Jn.
Jude	Jude	Jude
Revelation	Rev.	Rv.

- Range of chapters: `\bibleref{Mal. 3-4}`
- Range of verses: `\bibleref{Proverbs 3:5-6}`
- Range from a book to a chapter: `\bibleref{1 Chronicles - 2 Chronicles 9}`
- Range in a list within a chapter: `\bibleref{James 1:5-6,17}`
- Range in a list across chapters: `\bibleref{Ezra 7-10; Neh 8}`

`\setbiblestyle` The style of the rendered reference can be changed by calling `\setbiblestyle{`⟨*stylename*⟩`}`. The supplied styles, with an example reference, are shown in Table 2. Custom styles can also be added using the command `\providebiblestyle`, as shown in 3.8.5.

Table 2: Supported Styles

Style	**Example**
fullname	2 Timothy 3:16–17
jerusalem	2 Tm 3:16–17
anglosaxon	II Tim. III.16–17
JEH	2 Tim. iii. 16–17
NTG	2 Tim iii,16–17
MLA	2 Tim. iii.16–17
chicago	2 Tim. iii : 16–17
text	Second Epistle to Timothy, chapter three verse sixteen to verse seventeen
biblegateway.com	2 Timothy 3:16–17

3 Implementation

All the internal macros have a name beginning with `brm@`, which stands for biblerefmouth, so as to avoid polluting the global namespace.

It would have been nice if an expandable Bible reference could be created using the existing bibleref package, perhaps by adding `\protect` commands. Unfortunately, the reality of TEX is more complicated than that. The entire package needed to be reimplemented using a functional programming paradigm.

There are some techniques that need to be used over and over to perform complicated computations in TEX's mouth. All non-expandable primitives must be avoided. That means that assignments, (e.g. `\def`), cannot be used. Instead, recursion and currying and the command pattern are relied on extensively. In order to read this code, it doesn't hurt to be familiar with a language like Haskell.

3.1 Entry Points

`\setbiblestyle` The user can set which reference style is in effect with `\setbiblestyle{`⟨*stylename*⟩`}`. The available styles are defined in Table 2.

Custom styles can also be added using the command `\providebiblestyle`. This command just stores the style for later use by `\bibleref`.

Note that this command is itself *not* expandable, so it must be called outside of any expandable context. If you need to switch between different styles within an expandable context, you should use the `\brm@bibleref` command.

```
\newcommand*{\setbiblestyle}[1]{%
    \def\brm@currentstyle{#1}%
}
```

The default style for documents that don't override it is the one that outputs the full name of the books.

```
\setbiblestyle{fullname}
```

`\bibleref` This is the main entry point to the library. The user can specify any Bible reference, including lists and ranges, as the argument to `\bibleref{`⟨*reference*⟩`}`. Detailed usage instructions are given in section 2. This command just retrieves the current style and delegates to `\brm@bibleref`.

```
\newcommand*{\bibleref}[1]{%
    \brm@bibleref{\brm@currentstyle}{#1}%
}
```

`\brm@bibleref` The `\brm@bibleref{`⟨*stylename*⟩`}{`⟨*reference*⟩`}` command checks that the style exists, and then outputs a consistently formatted reference. It gets a list of ranges from `\brm@parseall`. Then it starts reading the list by creating a cons handler from the partially applied function `\brm@formatall`. The first argument passed to `\brm@formatall` is a style, which is itself a function with the signature ⟨*callback*⟩`{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}{`⟨*list*⟩`}`. No nil handler needs to be provided, because an empty list would be an error, and would not need to be formatted anyway.

```
\newcommand*{\brm@bibleref}[2]{%
    \brm@ifnameexists{\brm@style{#1}}{%
        \brm@readlist{\brm@parseall{#2}}{%
            \brm@formatall{\csname\brm@style{#1}\endcsname}%
        }{%
        }%
    }{%
        \brm@error{Unrecognized style: #1}%
    }%
}
```

`\brm@formatall` The formatting of a Bible reference starts with `\brm@formatall{`⟨*style*⟩`}{`⟨*range*⟩`}{`⟨*list*⟩`}`. This command gets called from `\brm@readlist`, so the last two arguments are the head and tail of a list. This command unpacks a range into a begin passage and an end passage, which are passed to the helper `\brm@applystyle`.

```
\newcommand*{\brm@formatall}[3]{%
    \brm@unpackrange{#2}{\brm@applystyle{#1}{#3}}%
}
```

`\brm@applystyle` The command `\brm@applystyle{`⟨*style*⟩`}{`⟨*list*⟩`}{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}` rearranges the arguments into the order that is recognized by the style function. It forwards the begin passage and end passage, and it also allows continued extraction from the list by passing the partially applied function `\brm@nextrange`. The style can finish applying `\brm@nextrange` by providing a style delegate and a nil handler. The style delegate that gets provided to format subsequent ranges need not be the same style command that is being applied to format the first range.

```
\newcommand*{\brm@applystyle}[4]{%
    #1{#3}{#4}{\brm@nextrange{#2}}%
}
```

`\brm@nextrange` The command `brm@nextrange{`⟨*list*⟩`}{`⟨*style*⟩`}{`⟨*nilhandler*⟩`}` allows styles to read the next range off of a list of references. Since the list will already have been given by `\brm@applystyle`, the style only needs to provide the style delegate and nil handler.

```
\newcommand*{\brm@nextrange}[3]{%
    \brm@readlist{#1}{%
        \brm@formatall{#2}%
    }{%
        #3%
    }%
}
```

3.2 Data Structures

A list can be represented as a recursive data structure, as in other functional languages. But data structures themselves are difficult to construct in TEX. Some hints can be found in the lazylist package. A data structure can be represented as a macro that accepts a callback. When the callback is invoked, all the data structure's fields are passed to it. The simplest form of callback is one that returns one field and ignores all the others. If that sounded complicated, then the following sample code might help or it might make the confusion worse.

`\brm@packpassage` One important data structure is the passage. It is created by `\brm@packpassage{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*verse*⟩`}`. You may notice that the command signature advertises 4 parameters, but the user only calls it with 3 arguments. That's because the fourth argument is a callback that will be supplied later, when it's time to extract the data. For now, all this command does with its arguments is passes them to the callback.

```
\newcommand*{\brm@packpassage}[4]{%
    #4{#1}{#2}{#3}%
}
```

`\thebook` The first such callback is `\brm@choosebook{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*verse*⟩`}`. It gets passed the three fields from the passage. It returns the book, and ignores the chapter and verse. There is also a public command, `\thebook{`⟨*passage*⟩`}`, which wraps

the operation in a command that has a more familiar syntax. These commands work in concert to support use cases like the following:

```
% Store a passage in \mypassage.
\def\mypassage{\brm@packpassage{Jeremiah}{16}{16}}
% Later, extract the book from \mypassage into \mybook.
\def\mybook{\thebook{\mypassage}}
```

```
\newcommand*{\thebook}[1]{%
    #1{\brm@choosebook}%
}
\newcommand*{\brm@choosebook}[3]{%
    #1%
}
```

`\thechapter` The `\thechapter{`⟨*passage*⟩`}` command is a similar command for extracting a chapter. The callback is `\brm@choosechapter`. It ignores the first and third fields, and returns the second field, which is the chapter.

```
\newcommand*{\thechapter}[1]{%
    #1{\brm@choosechapter}%
}
\newcommand*{\brm@choosechapter}[3]{%
    #2%
}
```

`\theverse` Likewise, `\theverse{`⟨*passage*⟩`}` extracts a verse. The callback `\brm@chooseverse` returns the third field and ignores the others.

```
\newcommand*{\theverse}[1]{%
    #1{\brm@chooseverse}%
}
\newcommand*{\brm@chooseverse}[3]{%
    #3%
}
```

`\brm@packrange` Another data structure is the range. It has a begin passage and and end passage. A range is created by calling `\brm@packrange{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}`.

```
\newcommand*{\brm@packrange}[3]{%
    #3{#1}{#2}%
}
```

`\brm@unpackrange` The corresponding command to extract the fields is `\brm@unpackrange{`⟨*range*⟩`}{`⟨*callback*⟩`}`. This doesn't follow exactly the same pattern as the passage helpers. Instead of having separate commands to extract the two fields, there is a single function that extracts both fields at the same time. Clients should pass in a callback with the signature ⟨*callback*⟩`{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}`.

```
\newcommand*{\brm@unpackrange}[2]{%
    #1{#2}%
}
```

`\brm@packseparators` Another data structure is for separators. This structure stores the delimiter characters that will be output between books and chapters, and between chapters and verses, in Bible references.

```
\newcommand*{\brm@packseparators}[3]{%
    #3{#1}{#2}%
}
```

`\bookchapterseparator` To extract the first field of the separators structure, use `\bookchapterseparator{`⟨*separators*⟩`}`. Instead of defining a custom callback, it was sufficient to use the existing `\@firstoftwo` command, which already does exactly what we want. It returns the first field and ignores the second field.

```
\newcommand*{\bookchapterseparator}[1]{%
    #1{\@firstoftwo}%
}
```

`\chapterverseseparator` To extract the second field of the separators structure, use `\chapterverseseparator{`⟨*separators*⟩`}`.

```
\newcommand*{\chapterverseseparator}[1]{%
    #1{\@secondoftwo}%
}
```

`\brm@nillist` Now we arrive at the package's only recursive data structure, which is for storing lists. The definition mirrors lists in the lamba calculus. This mechanism was suggested by the lazylist package. Each list node is a data structure that is either a nil or a node. A nil means an empty list. It is represented as a function that ignores its first argument and expands its second argument.

```
\newcommand*{\brm@nillist}{%
    \@secondoftwo%
}
```

`\brm@conslist` Every non-empty list node contains fields for the list head and the list tail. The list head is the current first item in the list. The list tail is its own list, and it contains all the items except the head. If the head is the only element in the list, then the tail is nil. A list node is represented as a function that calls its first argument and ignores its second argument. The first argument must be a callback with the signature ⟨*callback*⟩`{`⟨*head*⟩`}{`⟨*tail*⟩`}`.

```
\newcommand*{\brm@conslist}[2]{%
    \brm@conslist@choose{#1}{#2}%
}
\newcommand*{\brm@conslist@choose}[4]{%
    #3{#1}{#2}%
}
```

`\brm@readlist` Reading a list is done with the command `\brm@readlist{`⟨*list*⟩`}{`⟨*conshandler*⟩`}{`⟨*nilhandler*⟩`}`. If the list is empty, then the ⟨*nilhandler*⟩ will be called without any arguments. If the list is non-empty, then the ⟨*conshandler*⟩ will be called. It should have the signature ⟨*callback*⟩`{`⟨*head*⟩`}{`⟨*tail*⟩`}`.

```
\newcommand*{\brm@readlist}[3]{%
    #1{#2}{#3}%
}
```

`\brm@recordrange` A list of references is kept using the list structure. Each element is one of the range structures. The list is constructed with `\brm@recordrange{`⟨*beginbook*⟩`}{`⟨*beginchapter*⟩`}{`⟨*beginverse*⟩`}{`⟨*endbook*⟩`}{`⟨*endchapter*⟩`}{`⟨*endverse*⟩`}{`⟨*tail*⟩`}`. The tail passed to this command should be a list that contains all subsequent ranges, or nil if there are no more ranges.

```
\newcommand*{\brm@recordrange}[7]{%
    \brm@conslist{%
        \brm@packrange{%
         \brm@packpassage{#1}{#2}{#3}%
        }{%
         \brm@packpassage{#4}{#5}{#6}%
        }%
    }{%
        #7%
    }%
}
```

`\brm@recordpassage` A single passage can be added to the list as a range that has a regular begin passage and empty fields for the end passage. It is done with the command `\brm@recordpassage{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*verse*⟩`}{`⟨*tail*⟩`}`.

```
\newcommand*{\brm@recordpassage}[4]{%
    \brm@recordrange{#1}{#2}{#3}{}{}{}{#4}%
}
```

3.3 Parsing

3.3.1 Reference Parsing

`\brm@parseall` Parsing starts with the command `\brm@parseall{`⟨*remainingtext*⟩`}`. It checks whether the reference begins with a valid book. It finds the longest string at the beginning of the input that is a valid book, consuming both letters and numbers that are part of the book name. If it finds a match, then parsing continues with the `\brm@checkchapter` command.

Throughout this section, `1 Cor 15:20-22,40-42;16` will be used as a sample reference. The reference will be shown with a vertical bar, or cursor, at the point where the parsing has advanced. When this command is called, the cursor will be at: `⌊1 Cor 15:20-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@parseall}[1]{%
    \brm@parsebook{#1}{%
        \brm@checkchapter%
    }{%
        \brm@error{Unrecognized book: #1}%
    }%
```

```
}
```

`\brm@checkchapter` After the book has been parsed, `\brm@checkchapter{`⟨*book*⟩`}{`⟨*remainingtext*⟩`}` is called.

A valid reference may contain only a book name, in which case there is no remaining text.

The remaining text could also contain a chapter, which this command will check for. The last alternative is that there is a delimiter separating this reference from more references.

When this command is called, the cursor will be at: `1 Cor ⌊15:20-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@checkchapter}[2]{%
    \brm@ifempty{#2}{%
        \brm@recordpassage{#1}{}{}{\brm@nillist}%
    }{%
        \brm@ifsamehead{#2}{-}{%
            \brm@parserange{#1}{}{}{\brm@tailstr{#2}}%
        }{%
            \brm@ifsamehead{#2}{;}{%
                \brm@recordpassage{#1}{}{}{%
                    \brm@parseall{\brm@tailstr{#2}}%
                }%
            }{%
                \brm@parsechapter{#1}{#2}%
            }%
        }%
    }%
}
```

`\brm@parsechapter` After it has been determined that the next characters contain a chapter number, `\brm@parsechapter{`⟨*book*⟩`}{`⟨*remainingtext*⟩`}` will consume it. The `\brm@parsenumber` helper will pass the number and the remaining text to the `\brm@checkchapterdelim` callback.

When this command is called, the cursor will be at: `1 Cor ⌊15:20-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@parsechapter}[2]{%
    \brm@parsenumber{#2}{\brm@checkchapterdelim{#1}}%
}
```

`\brm@checkchapterdelim` After a chapter has been parsed, `\brm@checkchapterdelim{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*remainingtext*⟩`}` is called. If there are any characters remaining in the input, the next one should be a delimiter. There are several delimiters that would be valid at this point.

When this command is called, the cursor will be at: `1 Cor 15⌊:20-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@checkchapterdelim}[3]{%
    \brm@ifempty{#3}{%
        \brm@recordpassage{#1}{#2}{}{\brm@nillist}%
```

```
    }{%
        \brm@ifsamehead{#3}{:}{%
            \brm@parseverse{#1}{#2}{\brm@tailstr{#3}}%
        }{%
            \brm@ifsamehead{#3}{-}{%
                \brm@parserange{#1}{#2}{}{\brm@tailstr{#3}}%
            }{%
                \brm@ifsamehead{#3}{;}{%
                    \brm@recordpassage{#1}{#2}{}{%
                        \brm@parsechapterlist{#1}{\brm@tailstr{#3}}%
                    }%
                }{%
                    \brm@error{Unrecognized chapter trailer: #3}%
                }%
            }%
        }%
    }%
}
```

`\brm@parseverse` After a `:` delimiter, the next characters must be a verse number. The verse is parsed by `\brm@parseverse{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*remainingtext*⟩`}` The `\brm@parsenumber` helper will pass the number and the remaining text to the `\brm@checkversetrailer` callback.

When this command is called, the cursor will be at: `1 Cor 15:`⌊`20-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@parseverse}[3]{%
    \brm@parsenumber{#3}{\brm@checkversetrailer{#1}{#2}}%
}
```

`\brm@checkversetrailer` After the verse has been parsed, there may be some remaining input. It is passed to `\brm@checkversetrailer{`⟨*book*⟩`}{`⟨*chapter*⟩`}{`⟨*verse*⟩`}{`⟨*remainingtext*⟩`}`.

If there are no more characters, then parsing of the passage is done. If there are more characters, then the next one should be a delimiter. There are several delimiters that would be valid at this point.

When this command is called, the cursor will be at: `1 Cor 15:20`⌊`-1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@checkversetrailer}[4]{%
    \brm@ifempty{#4}{%
        \brm@recordpassage{#1}{#2}{#3}{\brm@nillist}%
    }{%
        \brm@ifsamehead{#4}{-}{%
            \brm@parserange{#1}{#2}{#3}{\brm@tailstr{#4}}%
        }{%
            \brm@ifsamehead{#4}{;}{%
                \brm@recordpassage{#1}{#2}{#3}{%
                    \brm@parsechapterlist{#1}{\brm@tailstr{#4}}%
                }%
            }{%
                \brm@ifsamehead{#4}{,}{%
```

```
                    \brm@recordpassage{#1}{#2}{#3}{%
                        \brm@parseverse{#1}{#2}{\brm@tailstr{#4}}%
                    }%
                }{%
                    \brm@error{Unrecognized verse trailer: #4}%
                }%
            }%
        }%
    }%
}
```

`\brm@parserange` After a `-` delimiter, the end passage of a range is parsed by `\brm@parserange{⟨beginbook⟩}{⟨beginchapter⟩}{⟨beginverse⟩}{⟨remainingtext⟩}`. The next characters could represent a book, chapter, or verse.

When this command is called, the cursor will be at: `1 Cor 15:20-`⌊`1 Cor 15:22,40-42;16`.

```
\newcommand*{\brm@parserange}[4]{%
    \brm@parsebook{#4}{%
        \brm@checkrangechapter{#1}{#2}{#3}%
    }{%
        \brm@ifempty{#2}{%
            \brm@error{Unrecognized range book: #4}%
        }{%
            \brm@parsenumber{#4}{%
                \brm@checkrangenumbertrailer{#1}{#2}{#3}{#1}%
            }%
        }%
    }%
}
```

`\brm@checkrangechapter` While the end passage of a range is being parsed, after a book has been found, the command `\brm@checkrangechapter{⟨beginbook⟩}{⟨beginchapter⟩}{⟨beginverse⟩}{⟨endbook⟩}{⟨remainingtext⟩}` looks for a subsequent chapter.

When this command is called, the cursor will be at: `1 Cor 15:20-1 Cor `⌊`15:22,40-42;16`.

```
\newcommand*{\brm@checkrangechapter}[5]{%
    \brm@ifempty{#5}{%
        \brm@recordrange{#1}{#2}{#3}{#4}{}{}{\brm@nillist}%
    }{%
        \brm@ifsamehead{#5}{;}{%
            \brm@recordrange{#1}{#2}{#3}{#4}{}{}{%
                \brm@parseall{\brm@tailstr{#5}}%
            }%
        }{%
            \brm@parsenumber{#5}{%
                \brm@checkrangechapterdelim{#1}{#2}{#3}{#4}%
            }%
        }%
    }%
```

```
}
```

`\brm@checkrangechapterdelim` While the end passage of a range is being parsed, after a book and chapter have been found, the command `\brm@checkrangechapterdelim{⟨beginbook⟩}{⟨beginchapter⟩}{⟨beginverse⟩}{⟨endbook⟩}{⟨endchapter⟩}{⟨remainingtext⟩}` is called. If there is any remaining text, it is expected to be a delimiter.

When this command is called, the cursor will be at: `1 Cor 15:20-1 Cor 15⌊:22,40-42;16`.

```
\newcommand*{\brm@checkrangechapterdelim}[6]{%
    \brm@ifempty{#6}{%
        \brm@recordrange{#1}{#2}{#3}{#4}{#5}{}{\brm@nillist}%
    }{%
        \brm@ifsamehead{#6}{:}{%
            \brm@parsenumber{\brm@tailstr{#6}}{%
                \brm@checkrangeversetrailer{#1}{#2}{#3}{#4}{#5}%
            }%
        }{%
            \brm@error{Unrecognized range chapter trailer: #6}%
        }%
    }%
}
```

`\brm@checkrangenumbertrailer` If a range starts with a number, then it can be ambiguous whether that number represents a chapter or verse. In that case, `\brm@checkrangenumbertrailer{⟨beginbook⟩}{⟨beginchapter⟩}{⟨beginverse⟩}{⟨endbook⟩}{⟨endnumber⟩}{⟨remainingtext⟩}` is called. The ambiguity can be resolved by checking whether there is a delimiter remaining, and which delimiter it is.

When this command is called, the cursor will be at: `1 Cor 15:20-1 Cor 15:22,40-42⌊;16`.

```
\newcommand*{\brm@checkrangenumbertrailer}[6]{%
    \brm@ifempty{#6}{%
        \brm@ifempty{#3}{%
            \brm@recordrange{#1}{#2}{}{#4}{#5}{}{\brm@nillist}%
        }{%
            \brm@recordrange{#1}{#2}{#3}{#4}{#2}{#5}{\brm@nillist}%
        }%
    }{%
        \brm@ifsamehead{#6}{:}{%
            \brm@parsenumber{\brm@tailstr{#6}}{%
                \brm@checkrangeversetrailer{#1}{#2}{#3}{#4}{#5}%
            }%
        }{%
            \brm@ifempty{#3}{%
                \brm@ifsamehead{#6}{;}{%
                    \brm@recordrange{#1}{#2}{}{#4}{#5}{}{%
                        \brm@parsechapterlist{#4}{\brm@tailstr{#6}}%
                    }%
                }{%
```

```
                    \brm@error{Unrecognized range number trailer: #6}%
                }%
            }{%
                \brm@ifsamehead{#6}{;}{%
                    \brm@recordrange{#1}{#2}{#3}{#4}{#2}{#5}{%
                        \brm@parsechapterlist{#4}{\brm@tailstr{#6}}%
                    }%
                }{%
                    \brm@ifsamehead{#6}{,}{%
                        \brm@recordrange{#1}{#2}{#3}{#4}{#2}{#5}{%
                            \brm@parseverse{#4}{#2}{\brm@tailstr{#6}}%
                        }%
                    }{%
                        \brm@error{Unrecognized range number trailer: #6}%
                    }%
                }%
            }%
        }%
    }%
}
```

`\brm@checkrangeversetrailer` After the begin and end passage of a range have both been parsed, `\brm@checkrangeversetrailer{`⟨*beginbook*⟩`}{`⟨*beginchapter*⟩`}{`⟨*beginverse*⟩`}{`⟨*endbook*⟩`}{`⟨*endchapter*⟩`}{`⟨*endverse*⟩`}{`⟨*remainingtext*⟩`}` is called. There may be remaining text for another reference. The next character is expected to be a delimiter.

When this command is called, the cursor will be at: `1 Cor 15:20-1 Cor 15:22`⌊`,40-42;16`.

```
\newcommand*{\brm@checkrangeversetrailer}[7]{%
    \brm@ifempty{#7}{%
        \brm@recordrange{#1}{#2}{#3}{#4}{#5}{#6}{\brm@nillist}%
    }{%
        \brm@ifsamehead{#7}{;}{%
            \brm@recordrange{#1}{#2}{#3}{#4}{#5}{#6}{%
                \brm@parsechapterlist{#4}{\brm@tailstr{#7}}%
            }%
        }{%
            \brm@ifsamehead{#7}{,}{%
                \brm@recordrange{#1}{#2}{#3}{#4}{#5}{#6}{%
                    \brm@parseverse{#4}{#5}{\brm@tailstr{#7}}%
                }%
            }{%
                \brm@error{Unrecognized verse trailer: #7}%
            }%
        }%
    }%
}
```

`\brm@parsechapterlist` If the previous range ended with a ; delimiter, then the next characters are known to be either a book or a chapter. The command

`\brm@parsechapterlist{`⟨*book*⟩`}{`⟨*remainingtext*⟩`}` will handle either case.

When this command is called, the cursor will be at: `1 Cor 15:20-1 Cor 15:22,40-42;`⌊`16`.

```
\newcommand*{\brm@parsechapterlist}[2]{%
    \brm@parsebook{#2}{%
        \brm@checkchapter%
    }{%
        \brm@parsechapter{#1}{#2}%
    }%
}
```

3.3.2 Book Name Parsing

`\brm@parsebook` The `\brm@parsebook{`⟨*text*⟩`}{`⟨*successhandler*⟩`}{`⟨*failurehandler*⟩`}` command parses a book name. A book is known to exist if the definition `\brm@bookalias@`⟨*book*⟩ exists. This command will find the longest string of the source that is a valid book name. In other words, it will check to see whether the whole string is a valid book name, and then it will try to match the whole string minus the last character, and so on recursively. Eventually it will either find a valid book or it will exhaust every possibility and fail. If it finds a valid book then it will call the success handler, which is expected to have the signature ⟨*callback*⟩`{`⟨*book*⟩`}{`⟨*remainingtext*⟩`}`. If it doesn't find a valid book it will call the failure handler with no arguments.

This command depends on `\brm@headstr`, which consumes all spaces in its input. That means that spaces in the text will be ignored. For example, if the text contains `Song of Songs`, then it will only match a book alias `SongofSongs`.

The real work is done by the `\brm@parsebook@accumulate` helper, so this method just delegates to it. It makes sure the text is fully-expanded before beginning. It passes an empty string as the accumulated text, because at the start of the process nothing has accumulated yet.

```
\newcommand*{\brm@parsebook}[3]{%
    \expandafter\brm@parsebook@accumulate\expandafter%
        {\romannumeral-`0#1}{}{#2}{#3}%
}
```

`\brm@parsebook@accumulate` The recursive workhorse for parsing books is `\brm@parsebook@accumulate{`⟨*remainingtext*⟩`}{`⟨*accumulatedtext*⟩`}{`⟨*successhandler*⟩`}{`⟨*unwindhandler*⟩`}`. The unwind handler is called with no arguments if the parsing fails.

This command uses `\brm@headstr` to pull the first token off of the remaining text string. It then adds that character to the end of the accumulated text. The new string is a potential book alias that has not been tested yet. So in that way, character by character, it is able to try every possible book name.

A complication is that it encounters each possible book name starting with the shortest, but it actually wants to give precedence to the longest book names. So when it finds a possible book name, it doesn't actually test it yet to see whether it is valid. Instead it simply makes a recursive call to try the next longer possible book name. It passes in an unwind handler, which will only be called if all the

longer book names fail. It is only within the unwind handler that tests whether the string is valid. If the string is valid, then the success handler is called, and parsing stops. If the string is invalid, then the caller's unwind handler will be invoked. In that way it will continue to unwind all the recursive calls, testing each successively shorter string at each level.

This code uses the `\romannumeral` trick to make sure that the text strings are always fully expanded, because otherwise the `\brm@tailstr` command would have to be re-evaluated many times on the same input. With the full expansions in place, this algorithm completes in linear time. Without them, it would be polynomial time.

```
\newcommand*{\brm@parsebook@accumulate}[4]{%
    \brm@ifempty{#1}{%
        #4%
    }{%
        \expandafter\brm@parsebook@expand\expandafter%
            {\romannumeral-‘0\brm@headstr{#1}}%
            {#2}%
            {\brm@tailstr{#1}}%
            {#3}%
            {%
                \brm@ifnameexists{\brm@bookalias{#2\brm@headstr{#1}}}{%
                    \expandafter\brm@parsebook@success\expandafter%
                        {\romannumeral-‘0\brm@tailstr{#1}}%
                        {\csname\brm@bookalias%
                            {#2\brm@headstr{#1}}\endcsname}%
                        {#3}%
                }{%
                    #4%
                }%
            }%
    }%
}
```

`\brm@parsebook@expand` The TEX syntax sometimes makes simple tasks difficult. In this case, it is difficult to fully expand two arguments to a command. The workaround is to expand one argument, pass everything to a helper, and then expand the other argument in the helper. The `\brm@parsebook@expand{`⟨*lastaccumulatedcharacter*⟩`}{`⟨*accumulatedtext*⟩`}{`⟨*remainingtext*⟩`}{`⟨*successhandler*⟩`}{`⟨*unwindhandler*⟩`}` command exists to expand the remaining text argument.

```
\newcommand*{\brm@parsebook@expand}[5]{%
    \expandafter\brm@parsebook@accumulate\expandafter%
        {\romannumeral-‘0#3}{#2#1}{#4}{#5}%
}
```

`\brm@parsebook@success` When a successful match with a book name has been made, the success handler is called. The remaining text needs to be expanded fully and then passed to the handler as the second argument. But it is difficult to fully expand any argument other than the first. As a workaround, the remaining text is fully expanded in

\brm@parsebook@accumulate, which passes it to this helper as the first argument. Then \brm@parsebook@success{⟨*remainingtext*⟩}{⟨*book*⟩}{⟨*successhandler*⟩} rearranges the arguments and calls the success handler.

```
\newcommand*{\brm@parsebook@success}[3]{%
    #3{#2}{#1}%
}
```

3.3.3 Number Parsing

\brm@parsenumber The \brm@parsenumber{⟨*text*⟩}{⟨*handler*⟩} command parses an integer. When it's finished, it calls the handler, which is expected to have the signature ⟨*callback*⟩{⟨*number*⟩}{⟨*remainingtext*⟩}. It delegates to the command command \brm@parsenumber@accumulate. It passes an empty string as the accumulated text, because at the start of the process nothing has accumulated yet.

This command depends on \brm@headstr, which consumes all spaces in its input. That means that spaces in the text will be ignored. For example, if the text contains `1 2 3`, then it will be parsed as if it were `123`.

```
\newcommand*{\brm@parsenumber}[2]{%
    \expandafter\brm@parsenumber@accumulate\expandafter%
        {\romannumeral-‘0#1}{}{#2}%
}
```

\brm@parsenumber@accumulate The work for parsing numbers is done in the \brm@parsenumber@accumulate{⟨*remainingtext*⟩}{⟨*accumulatedtext*⟩}{⟨*handler*⟩} command. It consumes the remaining text, one character at a time. As soon as it finds a character that is not a digit, it calls the handler.

```
\newcommand*{\brm@parsenumber@accumulate}[3]{%
    \brm@ifempty{#1}{%
        \brm@parsenumber@end{#1}{#2}{#3}%
    }{%
        \expandafter\brm@ifdigit\expandafter%
            {\romannumeral-‘0\brm@headstr{#1}}{%
            \expandafter\brm@parsenumber@expand\expandafter%
                {\romannumeral-‘0\brm@headstr{#1}}{\brm@tailstr{#1}}{#3}{#2}%
        }{%
            \brm@parsenumber@end{#1}{#2}{#3}%
        }%
    }%
}
```

\brm@parsenumber@expand It is difficult to fully expand two arguments to a command. The workaround is to expand one argument, pass everything to a helper, and then expand the other argument in the helper. The \brm@parsenumber@expand{⟨*lastaccumulatedcharacter*⟩}{⟨*accumulatedtext*⟩}{⟨*remainingtext*⟩}{⟨*handler*⟩} command exists to expand the remaining text argument.

```
\newcommand*{\brm@parsenumber@expand}[4]{%
    \expandafter\brm@parsenumber@accumulate\expandafter%
        {\romannumeral-‘0#2}{#4#1}{#3}%
```

```
}
```

`\brm@parsenumber@end` When the end of the number has been reached, the command `\brm@parsenumber@end{`⟨*remainingtext*⟩`}{`⟨*number*⟩`}{`⟨*handler*⟩`}` is called. It checks to make sure that at least some digits were parsed, and fails otherwise. Then it passes the number and remaining text to the handler, which is expected to have the signature ⟨*callback*⟩`{`⟨*number*⟩`}{`⟨*remainingtext*⟩`}`.

```
\newcommand*{\brm@parsenumber@end}[3]{%
    \brm@ifempty{#2}{%
        \brm@error{Invalid number: #1}%
    }{%
        #3{#2}{#1}%
    }%
}
```

3.4 Predicates

Each of the commands in this section is a predicate, which is a command that tests whether a certain property is true. These commands have both a true handler, which is called with no arguments if the condition is true, and a false handler, which is called if the condition is false.

`\brm@ifsamehead` The `\brm@ifsamehead{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given strings start with the same letter. It requires both strings to contain at least one character.

```
\newcommand*{\brm@ifsamehead}[4]{%
    \brm@ifsamechar{\brm@headstr{#1}}{\brm@headstr{#2}}{#3}{#4}%
}
```

`\brm@ifsamechar` The `\brm@ifsamechar{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given characters are the same.

```
\newcommand*{\brm@ifsamechar}[4]{%
    \if#1#2%
        \expandafter\@firstoftwo%
    \else%
        \expandafter\@secondoftwo%
    \fi%
    {#3}{#4}%
}
```

`\brm@ifsamestr` The `\brm@ifsamestr{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given strings are the same. Most implementations, such as `\ifstrequal`, are not expandable. This command checks the characters recursively one by one using `\if`.

```
\newcommand*{\brm@ifsamestr}[4]{%
    \expandafter\brm@ifsamestr@expandsecond\expandafter%
        {\romannumeral-‘0#2}{#1}{#3}{#4}%
}
```

`\brm@ifsamestr@expandsecond` It is difficult to fully expand two arguments to a command. The workaround is to expand one argument, pass everything to a helper, and then expand the other argument in the helper. The `\brm@ifsamestr@expandsecond{`⟨*right*⟩`}{`⟨*left*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command exists to expand the first string argument.

```
\newcommand*{\brm@ifsamestr@expandsecond}[4]{%
    \expandafter\brm@ifsamestr@expanded\expandafter%
        {\romannumeral-`0#2}{#1}{#3}{#4}%
}
```

`\brm@ifsamestr@expanded` After both the first and second strings have been expanded, equality is tested recursively by `\brm@ifsamestr@expanded{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}`. If both strings are empty, then they must be equal. If only one string is empty, then they are inequal. If neither of them are empty, then the first characters are checked for equality, and the rest are checked recursively.

```
\newcommand*{\brm@ifsamestr@expanded}[4]{%
    \brm@ifempty{#1}{%
        \brm@ifempty{#2}{%
            #3%
        }{%
            #4%
        }%
    }{%
        \brm@ifempty{#2}{%
            #4%
        }{%
            \brm@ifsamehead{#1}{#2}{%
                \brm@ifsamestr{\brm@tailstr{#1}}{\brm@tailstr{#2}}{%
                    #3%
                }{%
                    #4%
                }%
            }{%
                #4%
            }%
        }%
    }%
}
```

`\ifsamebook` The `\ifsamebook{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given books are the same.

```
\newcommand*{\ifsamebook}[4]{%
    \brm@ifsamestr{\thebook{#1}}{\thebook{#2}}{#3}{#4}%
}
```

`\ifsamechapter` The `\ifsamechapter{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given chapters are the same.

```
\newcommand*{\ifsamechapter}[4]{%
    \brm@ifsamestr{\thechapter{#1}}{\thechapter{#2}}{#3}{#4}%
}
```

`\ifsameverse` The `\ifsameverse{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the two given verses are the same.

```
\newcommand*{\ifsameverse}[4]{%
    \brm@ifsamestr{\theverse{#1}}{\theverse{#2}}{#3}{#4}%
}
```

`\ifhasbook` The `\ifhasbook{`⟨*passage*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the passage contains a book.

```
\newcommand*{\ifhasbook}[3]{%
    \expandafter\brm@ifempty\expandafter%
        {\romannumeral-`0\thebook{#1}}{#3}{#2}%
}
```

`\ifhaschapter` The `\ifhaschapter{`⟨*passage*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the passage contains a chapter.

```
\newcommand*{\ifhaschapter}[3]{%
    \expandafter\brm@ifempty\expandafter%
        {\romannumeral-`0\thechapter{#1}}{#3}{#2}%
}
```

`\ifhasverse` The `\ifhasverse{`⟨*passage*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the passage contains a verse.

```
\newcommand*{\ifhasverse}[3]{%
    \expandafter\brm@ifempty\expandafter%
        {\romannumeral-`0\theverse{#1}}{#3}{#2}%
}
```

`\brm@ifempty` The `\brm@ifempty{`⟨*text*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the string is empty. This test was suggested at `http://tex.stackexchange.com/questions/2936/test-whether-token-list-is-empty`.

```
\newcommand*{\brm@ifempty}[3]{%
    \expandafter\ifx\expandafter\brm@nevermatch#1\brm@nevermatch%
        \expandafter\@firstoftwo%
    \else%
        \expandafter\@secondoftwo%
    \fi%
    {#2}{#3}%
}
```

`\brm@nevermatch` The `\brm@nevermatch` command expands to itself. Because of that it is a convenient token to use in the `\brm@ifempty` test.

```
\def\brm@nevermatch{\brm@nevermatch}
```

`\brm@ifnameexists` The `\brm@ifnameexists{`⟨*csname*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the named is a defined command.

```
\newcommand*{\brm@ifnameexists}[3]{%
    \ifcsname#1\endcsname%
        \expandafter\@firstoftwo%
    \else%
        \expandafter\@secondoftwo%
    \fi%
    {#2}{#3}%
}
```

`\brm@ifdigit` The `\brm@ifdigit{`⟨*text*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the given character is a digit.

```
\newcommand*{\brm@ifdigit}[3]{%
    \ifnum 9<1\string#1 %
        \expandafter\@firstoftwo%
    \else%
        \expandafter\@secondoftwo%
    \fi%
    {#2}{#3}%
}
```

`\brm@ifspace` The `\brm@ifspace{`⟨*text*⟩`}{`⟨*truehandler*⟩`}{`⟨*falsehandler*⟩`}` command checks whether the given character is a space.

```
\newcommand*{\brm@ifspace}[3]{%
    \ifcat #1 %
        \expandafter\@firstoftwo%
    \else%
        \expandafter\@secondoftwo%
    \fi%
    {#2}{#3}%
}
```

3.5 String Manipulations

The commands in this section are expandable versions of common string manipulations. They work well for the use cases that exist in this package, but they tend to behave badly if they are given anything other than ASCII characters.

`\brm@uppercaseascii` The `\brm@uppercaseascii{`⟨*text*⟩`}` command changes an ASCII string to uppercase. Most implementations, such as `\MakeUppercase`, are not expandable. The limitation to ASCII characters here is not serious, since an expandable implementation is only needed when a command name is being defined, and command names can only contain ASCII characters anyway.

```
\newcommand*{\brm@uppercaseascii}[1]{%
    \brm@ifempty{#1}{}{%
        \expandafter\brm@uppercaseascii@char\expandafter%
            {\romannumeral-`0\brm@headstr{#1}}%
        \expandafter\brm@uppercaseascii\expandafter%
            {\romannumeral-`0\brm@tailstr{#1}}%
    }%
```

```
}
```

`\brm@uppercaseascii@char`

The `\brm@uppercaseascii@char{`⟨*character*⟩`}` command changes a single ASCII character to uppercase. It uses a giant `\ifcase` statement to separate the lowercase letters from all other characters.

```
\newcommand*{\brm@uppercaseascii@char}[1]{%
    \ifcase`#1%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        #1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or#1\or%
        A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or K\or L\or M\or%
        N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or Y\or Z%
    \else#1%
    \fi%
}
```

`\brm@headstr`

The `\brm@headstr{`⟨*text*⟩`}` command returns the first character from a string. The string is required to contain at least one character, so callers should check `\brm@ifempty` first.

Because of the way it pulls the first character off the string, spaces all get consumed by the TeXparser. It is as if all spaces in the text get ignored. So really, this command returns the first non-space character in the text. Unfortunately, this is a deficiency of TeXthat cannot be worked around in an expandable manner. Even modifying the catcode of the space character does not fix this, because catcode changes and `\scantokens` are not themselves expandable. *If you know how to correct this problem, by writing an expandable function that returns the first character, whether or not it is a space, please contact the author.*

```
\def\brm@headstr#1{\brm@headstr@helper#1\@nil}
\def\brm@headstr@helper#1#2\@nil{#1}
```

`\brm@tailstr`

The `\brm@tailstr{`⟨*text*⟩`}` command returns the string with the first character removed. The string is required to contain at least one character, so callers should check `\brm@ifempty` first.

```
\def\brm@tailstr#1{\brm@tailstr@helper#1\@nil}
\def\brm@tailstr@helper#1#2\@nil{#2}
```

3.6 Exceptions

`\brm@error`

The `\brm@error{`⟨*message*⟩`}` command raises errors to the user. This is useful because other implementations, such as `\PackageError`, are not expandable. This implementation was suggested at `http://tex.stackexchange.com/questions/17238/signalling-error-in-expandable-context`.

```
\begingroup
\catcode`\:=11\relax
\gdef\brm@error#1{%
    \protect\expandafter\brm@error@aux\csname#1\endcsname%
}
\gdef\brm@error@aux#1{%
    \romannumeral-`0\@firstofone{\brm@error: #1}%
}
\endgroup
```

3.7 Formatting

These commands are used to format the reference according to a customizable style. Those commands that are used to customized styles are defined without the `brm@` prefix, to make them easier to access.

`\brm@bookref` The `\brm@bookref{`⟨*stylename*⟩`}{`⟨*book*⟩`}` command returns the name of the command that outputs the stylized book name.

```
\newcommand*{\brm@bookref}[2]{brm@bookref@#1@#2}
```

`\brm@bookalias` The `\brm@bookalias{`⟨*book*⟩`}` command returns the name of the command that outputs the actual book name for the book alias.

```
\newcommand*{\brm@bookalias}[1]{brm@bookalias@#1}
```

`\brm@style` The `\brm@style{`⟨*stylename*⟩`}` command returns the name of the style command that formats the reference.

```
\newcommand*{\brm@style}[1]{brm@style@#1}
```

`\thebookname` The `\thebookname{`⟨*stylename*⟩`}{`⟨*passage*⟩`}` outputs the stylized book name for the passage.

```
\newcommand*{\thebookname}[2]{%
    \brm@bookname{#1}{\thebook{#2}}%
}
```

`\brm@bookname` The `\brm@bookname{`⟨*stylename*⟩`}{`⟨*passage*⟩`}` outputs the stylized name for the given book.

```
\newcommand*{\brm@bookname}[2]{%
    \brm@ifnameexists{\brm@bookref{#1}{#2}}{%
        \csname\brm@bookref{#1}{#2}\endcsname%
    }{%
        \brm@error{Style missing book: #2}%
    }%
}
```

`\providebiblestyle` To create a new Bible style, the user calls `\providebiblestyle{`⟨*stylename*⟩`}{`⟨*formatter*⟩`}`. This command creates a style with the signature `\brm@style@`⟨*stylename*⟩`{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}{`⟨*list*⟩`}`. The formatter gets expanded from within an environment where those three arguments are defined.

```
\newcommand*{\providebiblestyle}[2]{%
    \expandafter\def\csname\brm@style{#1}\endcsname##1##2##3{%
        #2%
    }%
}
```

`\providebiblebookalias` A new Bible book alias can be created with `\providebiblebookalias{`⟨*book*⟩`}{`⟨*alias*⟩`}`. An alias is the name that is specified to the `\bibleref` command. Every book needs to have aliases defined for users to refer to it. A book can have as many different aliases as the user wishes to defined. This command expands the alias name and delegates to `\brm@providebiblebookalias`.

Because the parser ignores spaces, an alias name also cannot contain spaces. If an alias is created that contains spaces anyway, then it will be impossible for users to reference it.

```
\newcommand*{\providebiblebookalias}[2]{%
    \expandafter\brm@providebiblebookalias\expandafter%
        {\romannumeral-`0#2}{#1}%
}
```

`\brm@providebiblebookalias` A new Bible book alias is created by `\brm@providebiblebookalias{`⟨*alias*⟩`}{`⟨*book*⟩`}`.

```
\newcommand*{\brm@providebiblebookalias}[2]{%
    \expandafter\def\csname\brm@bookalias{#1}\endcsname{#2}%
}
```

`\providebiblebook` A new stylized Bible book name can be created with `\providebiblebook{`⟨*stylename*⟩`}{`⟨*book*⟩`}{`⟨*bookname*⟩`}`. The book name needs to be defined for each book and each style before it can be referenced by a user.

```
\newcommand*{\providebiblebook}[3]{%
    \expandafter\def\csname\brm@bookref{#1}{#2}\endcsname{#3}%
}
```

`\brm@formatchapter` This helper command extracts a chapter from a passage and then formats it according to the given callback. The signature is `\brm@formatchapter{`⟨*formatter*⟩`}{`⟨*passage*⟩`}`. It is used by partially applying the command with a formatter, creating a command that accept a passage and output a stylized chapter. The callback is expected to have the signature ⟨`\`*callback*⟩`{`⟨*chapter*⟩`}`.

```
\newcommand*{\brm@formatchapter}[2]{%
    #1{\thechapter{#2}}%
}
```

`\brm@formatverse` This helper command extracts a verse from a passage and then formats it according to the given callback. The signature is `\brm@formatverse{`⟨*formatter*⟩`}{`⟨*passage*⟩`}`. It is used by partially applying the command with a formatter, creating a command that accept a passage

and output a stylized verse. The callback is expected to have the signature ⟨*callback*⟩{⟨*verse*⟩}.

```
\newcommand*{\brm@formatverse}[2]{%
    #1{\theverse{#2}}%
}
```

`\brm@identity@filter` The `\brm@standard@filtered` command accepts a filter, but some callers don't need one. To meet that need, callers can use this filter that doesn't modify it's arguments, `\brm@identity@filter`{⟨*lastpassage*⟩}{⟨*passage*⟩}{⟨*infixseparators*⟩}{⟨*prefixes*⟩}{⟨*formatters*⟩}{⟨*delegate*⟩}.

```
\newcommand*{\brm@identity@filter}[6]{%
    #6{#1}{#2}{#3}{#4}{#5}%
}
```

`\standardbiblestyle` Many different Bible styles have a lot of similarities in structure. As long as the stylized references follow a standard pattern, they can be formatted with this command. There are more than nine parameters that are needed, and TEXonly allows nine parameters to any one command. The workaround is to consume the first nine arguments in this command and then consume the rest of the arguments in `\brm@standard@filtered`. The signature, including all parameters, is `\standardbiblestyle`{⟨*bookstyle*⟩}{⟨*bkchsep*⟩}{⟨*chvsep*⟩}{⟨*bksep*⟩}{⟨*chsep*⟩}{⟨*vsep*⟩}{⟨*rangesep*⟩}{⟨*chapterstyle*⟩}{⟨*versestyle*⟩}{⟨*beginpassage*⟩}{⟨*endpassage*⟩}{⟨*list*⟩}.

```
\newcommand*{\standardbiblestyle}[9]{%
    \brm@standard@filtered%
        {\brm@packseparators{#2}{#3}}%
        {\brm@packpassage{#4}{#5}{#6}}%
        {\brm@packpassage{#7}{#7}{#7}}%
        {\brm@packpassage%
            {\thebookname{#1}}%
            {\brm@formatchapter{#8}}%
            {\brm@formatverse{#9}}}%
        {\brm@identity@filter}%
}
```

`\brm@standard@filtered` This command helps collect arguments by consuming five arguments from `\standardbiblestyle` and three remaining arguments for the range begin, range end, and the remaining ranges list. It's signature is `\brm@standard@filtered`{⟨*infixseparators*⟩}{⟨*listseparators*⟩}{⟨*rangeseparators*⟩}{⟨*formatters*⟩}{⟨*filter*⟩}{⟨*beginpassage*⟩}{⟨*endpassage*⟩}{⟨*list*⟩}.

```
\newcommand*{\brm@standard@filtered}[8]{%
    #5%
        {\brm@packpassage{}{}{}}%
        {#6}%
        {#1}%
        {\brm@packpassage{}{}{}}%
        {#4}%
        {\brm@standard@passage}%
```

```
    #5{#6}{#7}{#1}{#3}{#4}{\brm@standard@passage}%
    #8{%
        \brm@standard@list{#1}{#2}{#3}{#4}{#5}%
            {\ifhasbook{#7}{#7}{#6}}%
    }{%
    }%
}
```

`\brm@standard@list` A standard list of references can be formatted by `\brm@standard@list{`⟨*infixseparators*⟩`}{`⟨*listseparators*⟩`}{`⟨*rangeseparators*⟩`}{`⟨*formatters*⟩`}{`⟨*filter*⟩`}{`⟨*lastpassage*⟩`}{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}{`⟨*list*⟩`}`. It delegates to `\brm@standard@passage` to format passages and then recursively calls itself to continue with the next element in the list.

```
\newcommand*{\brm@standard@list}[9]{%
    #5{#6}{#7}{#1}{#2}{#4}{\brm@standard@passage}%
    #5{#7}{#8}{#1}{#3}{#4}{\brm@standard@passage}%
    #9{%
        \brm@standard@list{#1}{#2}{#3}{#4}{#5}%
            {\ifhasbook{#8}{#8}{#7}}%
    }{%
    }%
}
```

`\brm@standard@passage` A single standard passage format is created by `\brm@standard@passage{`⟨*lastpassage*⟩`}{`⟨*passage*⟩`}{`⟨*infixseparators*⟩`}{`⟨*prefixes*⟩`}{`⟨*formatters*⟩`}`. The last passage is not displayed, but it is used to decide which separators might be necessary, and how much of the reference to output. For example, if the last passage is for a different chapter in the same book compared to the current passage, then the book name will not be output, but a prefix separator for chapters will be. This function contains an enormous number of nested "if-then" blocks to cover every possible relationship between the last passage and the current passage.

```
\newcommand*{\brm@standard@passage}[5]{%
    \ifhasbook{#2}{%
        \ifhaschapter{#2}{%
            \ifhasverse{#2}{%
                \ifsamebook{#1}{#2}{%
                    \ifsamechapter{#1}{#2}{%
                        \ifsameverse{#1}{#2}{%
                        }{%
                            \ifhasverse{#1}{%
                                \theverse{#4}%
                                \theverse{#5}{#2}%
                            }{%
                                \thechapter{#4}%
                                \thechapter{#5}{#2}%
                                \chapterverseseparator{#3}%
                                \theverse{#5}{#2}%
                            }%
                        }%
```

```
                }{%
                    \ifhaschapter{#1}{%
                        \thechapter{#4}%
                        \thechapter{#5}{#2}%
                        \chapterverseseparator{#3}%
                        \theverse{#5}{#2}%
                    }{%
                        \thebook{#4}%
                        \thebook{#5}{#2}%
                        \bookchapterseparator{#3}%
                        \thechapter{#5}{#2}%
                        \chapterverseseparator{#3}%
                        \theverse{#5}{#2}%
                    }%
                }%
            }{%
                \thebook{#4}%
                \thebook{#5}{#2}%
                \bookchapterseparator{#3}%
                \thechapter{#5}{#2}%
                \chapterverseseparator{#3}%
                \theverse{#5}{#2}%
            }%
        }{%
            \ifsamebook{#1}{#2}{%
                \ifsamechapter{#1}{#2}{%
                    \ifhasverse{#1}{%
                        \brm@ifsamestr
                            {\thechapter{#4}}{\theverse{#4}}{%
                            \thebook{#4}%
                            \thebook{#5}{#2}%
                            \bookchapterseparator{#3}%
                            \thechapter{#5}{#2}%
                        }{%
                            \thechapter{#4}%
                            \thechapter{#5}{#2}%
                        }%
                    }{%
                    }%
                }{%
                    \ifhaschapter{#1}{%
                        \ifhasverse{#1}{%
                            \brm@ifsamestr
                                {\thechapter{#4}}{\theverse{#4}}{%
                                \thebook{#4}%
                                \thebook{#5}{#2}%
                                \bookchapterseparator{#3}%
                                \thechapter{#5}{#2}%
                            }{%
                                \thechapter{#4}%
```

```
                                    \thechapter{#5}{#2}%
                                }%
                            }{%
                                \thechapter{#4}%
                                \thechapter{#5}{#2}%
                            }%
                        }{%
                            \thebook{#4}%
                            \thebook{#5}{#2}%
                            \bookchapterseparator{#3}%
                            \thechapter{#5}{#2}%
                        }%
                    }%
                }{%
                    \thebook{#4}%
                    \thebook{#5}{#2}%
                    \bookchapterseparator{#3}%
                    \thechapter{#5}{#2}%
                }%
            }%
        }{%
            \ifsamebook{#1}{#2}{%
                \ifhaschapter{#1}{%
                    \thebook{#4}%
                    \thebook{#5}{#2}%
                }{%
                }%
            }{%
                \thebook{#4}%
                \thebook{#5}{#2}%
            }%
        }%
    }{%
    }%
}
```

`\brm@delegatestyle` For standard formats that want to internally delegate part of the format to a different defined style, the `\brm@delegatestyle{`⟨*stylename*⟩`}{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}{`⟨*list*⟩`}` command is convenient.

```
\newcommand*{\brm@delegatestyle}[4]{%
    \brm@ifnameexists{\brm@style{#1}}{%
        \csname\brm@style{#1}\endcsname{#2}{#3}{#4}%
    }{%
        \brm@error{Unrecognized style: #1}%
    }%
}
```

3.8 Standard Styles

3.8.1 Book Names

All the same books and book names that are supported by the bibleref package are supported here too. Each book is given a full name, an abbreviation, and an alternative abbrevation. Users can define more names or abbreviations if they wish. Examples of how to extend the collection of supported books are shown in the bibleref-lds package.

`\brm@foreachbook@full` There are several different use cases for full book names. There are also several varieties of each full name. For example, the book number could be formatted with either Arabic or Roman numerals. To provide the greatest amount of flexibility, callback gets invoked for each formatted full name. This command has the signature `\brm@foreachbook@full{`⟨*callback*⟩`}{`⟨*space*⟩`}{`⟨*bookof*⟩`}{`⟨*gospelof*⟩`}{`⟨*epistleto*⟩`}{`⟨*epistletothe*⟩`}{`⟨*epistleof*⟩`}{`⟨*booknumberstyle*⟩`}{`⟨*epistlenumberstyle*⟩`}`. The callback argument is expected to have the signature ⟨*callback*⟩`{`⟨*book*⟩`}{`⟨*bookname*⟩`}`. The callback can do whatever it wants with the names, such as creating a permanant book alias. The two number style arguments are expected to be functions with the signature ⟨*callback*⟩`{`⟨*number*⟩`}`.

```
\newcommand*{\brm@foreachbook@full}[9]{%
    #1{Genesis}{#3Genesis}%
    #1{Exodus}{#3Exodus}%
    #1{Leviticus}{#3Leviticus}%
    #1{Numbers}{#3Numbers}%
    #1{Deuteronomy}{#3Deuteronomy}%
    #1{Joshua}{#3Joshua}%
    #1{Judges}{#3Judges}%
    #1{Ruth}{#3Ruth}%
    #1{ISamuel}{#8{1}#3Samuel}%
    #1{IISamuel}{#8{2}#3Samuel}%
    #1{IKings}{#8{1}#3Kings}%
    #1{IIKings}{#8{2}#3Kings}%
    #1{IChronicles}{#8{1}#3Chronicles}%
    #1{IIChronicles}{#8{2}#3Chronicles}%
    #1{Ezra}{#3Ezra}%
    #1{Nehemiah}{#3Nehemiah}%
    #1{Tobit}{#3Tobit}%
    #1{Judith}{#3Judith}%
    #1{Esther}{#3Esther}%
    #1{IMaccabees}{#8{1}#3Maccabees}%
    #1{IIMaccabees}{#8{2}#3Maccabees}%
    #1{Job}{#3Job}%
    #1{Psalms}{#3Psalms}%
    #1{Proverbs}{#3Proverbs}%
    #1{Ecclesiastes}{#3Ecclesiastes}%
    #1{SongofSongs}{#3Song#2of#2Songs}%
    #1{Wisdom}{#3Wisdom}%
```

```
    #1{Ecclesiasticus}{#3Ecclesiasticus}%
    #1{Isaiah}{#3Isaiah}%
    #1{Jeremiah}{#3Jeremiah}%
    #1{Lamentations}{#3Lamentations}%
    #1{Baruch}{#3Baruch}%
    #1{Ezekiel}{#3Ezekiel}%
    #1{Daniel}{#3Daniel}%
    #1{Hosea}{#3Hosea}%
    #1{Joel}{#3Joel}%
    #1{Amos}{#3Amos}%
    #1{Obadiah}{#3Obadiah}%
    #1{Jonah}{#3Jonah}%
    #1{Micah}{#3Micah}%
    #1{Nahum}{#3Nahum}%
    #1{Habakkuk}{#3Habakkuk}%
    #1{Zephaniah}{#3Zephaniah}%
    #1{Haggai}{#3Haggai}%
    #1{Zechariah}{#3Zechariah}%
    #1{Malachi}{#3Malachi}%
    #1{Matthew}{#4Matthew}%
    #1{Mark}{#4Mark}%
    #1{Luke}{#4Luke}%
    #1{John}{#4John}%
    #1{Acts}{Acts}%
    #1{Romans}{#6Romans}%
    #1{ICorinthians}{#9{1}#6Corinthians}%
    #1{IICorinthians}{#9{2}#6Corinthians}%
    #1{Galatians}{#6Galatians}%
    #1{Ephesians}{#6Ephesians}%
    #1{Philippians}{#6Philippians}%
    #1{Colossians}{#6Colossians}%
    #1{IThessalonians}{#9{1}#6Thessalonians}%
    #1{IIThessalonians}{#9{2}#6Thessalonians}%
    #1{ITimothy}{#9{1}#5Timothy}%
    #1{IITimothy}{#9{2}#5Timothy}%
    #1{Titus}{#5Titus}%
    #1{Philemon}{#5Philemon}%
    #1{Hebrews}{#6Hebrews}%
    #1{James}{#7James}%
    #1{IPeter}{#9{1}#7Peter}%
    #1{IIPeter}{#9{2}#7Peter}%
    #1{IJohn}{#9{1}#7John}%
    #1{IIJohn}{#9{2}#7John}%
    #1{IIIJohn}{#9{3}#7John}%
    #1{Jude}{#7Jude}%
    #1{Revelation}{#3Revelation}%
}
```

`\brm@foreachbook@abbrv` The abbreviated book names are accessible through `\brm@foreachbook@abbrv{⟨callback⟩}{⟨period⟩}{⟨booknumberstyle⟩}{⟨epistlenumberstyle⟩}` The callback ar-

gument is expected to have the signature ⟨*callback*⟩{⟨*book*⟩}{⟨*bookname*⟩}. The two number style arguments are expected to be functions with the signature ⟨*callback*⟩{⟨*number*⟩}.

```
\newcommand*{\brm@foreachbook@abbrv}[4]{%
    #1{Genesis}{Gn#2}%
    #1{Exodus}{Ex#2}%
    #1{Leviticus}{Lv#2}%
    #1{Numbers}{Nb#2}%
    #1{Deuteronomy}{Dt#2}%
    #1{Joshua}{Jos#2}%
    #1{Judges}{Jg#2}%
    #1{Ruth}{Rt#2}%
    #1{ISamuel}{#3{1}S#2}%
    #1{IISamuel}{#3{2}S#2}%
    #1{IKings}{#3{1}K#2}%
    #1{IIKings}{#3{2}K#2}%
    #1{IChronicles}{#3{1}Ch#2}%
    #1{IIChronicles}{#3{2}Ch#2}%
    #1{Ezra}{Ezr#2}%
    #1{Nehemiah}{Ne#2}%
    #1{Tobit}{Tb#2}%
    #1{Judith}{Jdt#2}%
    #1{Esther}{Est#2}%
    #1{IMaccabees}{#3{1}M#2}%
    #1{IIMaccabees}{#3{2}M#2}%
    #1{Job}{Jb#2}%
    #1{Psalms}{Ps#2}%
    #1{Proverbs}{Pr#2}%
    #1{Ecclesiastes}{Qo#2}%
    #1{SongofSongs}{Sg#2}%
    #1{Wisdom}{Ws#2}%
    #1{Ecclesiasticus}{Si#2}%
    #1{Isaiah}{Is#2}%
    #1{Jeremiah}{Jr#2}%
    #1{Lamentations}{Lm#2}%
    #1{Baruch}{Ba#2}%
    #1{Ezekiel}{Ezk#2}%
    #1{Daniel}{Dn#2}%
    #1{Hosea}{Ho#2}%
    #1{Joel}{Jl#2}%
    #1{Amos}{Am#2}%
    #1{Obadiah}{Ob#2}%
    #1{Jonah}{Jon#2}%
    #1{Micah}{Mi#2}%
    #1{Nahum}{Na#2}%
    #1{Habakkuk}{Hab#2}%
    #1{Zephaniah}{Zp#2}%
    #1{Haggai}{Hg#2}%
    #1{Zechariah}{Zc#2}%
```

```
    #1{Malachi}{Ml#2}%
    #1{Matthew}{Mt#2}%
    #1{Mark}{Mk#2}%
    #1{Luke}{Lk#2}%
    #1{John}{Jn#2}%
    #1{Acts}{Ac#2}%
    #1{Romans}{Rm#2}%
    #1{ICorinthians}{#4{1}Co#2}%
    #1{IICorinthians}{#4{2}Co#2}%
    #1{Galatians}{Ga#2}%
    #1{Ephesians}{Ep#2}%
    #1{Philippians}{Ph#2}%
    #1{Colossians}{Col#2}%
    #1{IThessalonians}{#4{1}Th#2}%
    #1{IIThessalonians}{#4{2}Th#2}%
    #1{ITimothy}{#4{1}Tm#2}%
    #1{IITimothy}{#4{2}Tm#2}%
    #1{Titus}{Tt#2}%
    #1{Philemon}{Phm#2}%
    #1{Hebrews}{Heb#2}%
    #1{James}{Jm#2}%
    #1{IPeter}{#4{1}P#2}%
    #1{IIPeter}{#4{2}P#2}%
    #1{IJohn}{#4{1}Jn#2}%
    #1{IIJohn}{#4{2}Jn#2}%
    #1{IIIJohn}{#4{3}Jn#2}%
    #1{Jude}{Jude}%
    #1{Revelation}{Rv#2}%
}
```

`\brm@foreachbook@altabbrv` The alternative abbreviated book names are accessible through `\brm@foreachbook@altabbrv{`⟨*callback*⟩`}{`⟨*period*⟩`}{`⟨*space*⟩`}{`⟨*booknumberstyle*⟩`}{`⟨*epistlenumberstyle*⟩`}` The callback argument is expected to have the signature ⟨*callback*⟩`{`⟨*book*⟩`}{`⟨*bookname*⟩`}`. The two number style arguments are expected to be functions with the signature ⟨*callback*⟩`{`⟨*number*⟩`}`.

```
\newcommand*{\brm@foreachbook@altabbrv}[5]{%
    #1{Genesis}{Gen#2}%
    #1{Exodus}{Exod#2}%
    #1{Leviticus}{Lev#2}%
    #1{Numbers}{Num#2}%
    #1{Deuteronomy}{Deut#2}%
    #1{Joshua}{Josh#2}%
    #1{Judges}{Judg#2}%
    #1{Ruth}{Ruth}%
    #1{ISamuel}{#4{1}Sam#2}%
    #1{IISamuel}{#4{2}Sam#2}%
    #1{IKings}{#4{1}Kgs#2}%
    #1{IIKings}{#4{2}Kgs#2}%
    #1{IChronicles}{#4{1}Chr#2}%
```

```
    #1{IIChronicles}{#4{2}Chr#2}%
    #1{Ezra}{Ezra}%
    #1{Nehemiah}{Neh#2}%
    #1{Tobit}{Tobit}%
    #1{Judith}{Judith}%
    #1{Esther}{Esther}%
    #1{IMaccabees}{#4{1}M#2}%
    #1{IIMaccabees}{#4{2}M#2}%
    #1{Job}{Job}%
    #1{Psalms}{Ps#2}%
    #1{Proverbs}{Prov#2}%
    #1{Ecclesiastes}{Eccles#2}%
    #1{SongofSongs}{S#2#3of#3S#2}%
    #1{Wisdom}{Wisd#2}%
    #1{Ecclesiasticus}{Ecclus#2}%
    #1{Isaiah}{Isa#2}%
    #1{Jeremiah}{Jer#2}%
    #1{Lamentations}{Lam#2}%
    #1{Baruch}{Baruch}%
    #1{Ezekiel}{Ezek#2}%
    #1{Daniel}{Dan#2}%
    #1{Hosea}{Hos#2}%
    #1{Joel}{Joel}%
    #1{Amos}{Amos}%
    #1{Obadiah}{Obad#2}%
    #1{Jonah}{Jonah}%
    #1{Micah}{Mic#2}%
    #1{Nahum}{Nah#2}%
    #1{Habakkuk}{Hab#2}%
    #1{Zephaniah}{Zeph#2}%
    #1{Haggai}{Hag#2}%
    #1{Zechariah}{Zech#2}%
    #1{Malachi}{Mal#2}%
    #1{Matthew}{Matt#2}%
    #1{Mark}{Mark}%
    #1{Luke}{Luke}%
    #1{John}{John}%
    #1{Acts}{Acts}%
    #1{Romans}{Rom#2}%
    #1{ICorinthians}{#5{1}Cor#2}%
    #1{IICorinthians}{#5{2}Cor#2}%
    #1{Galatians}{Gal#2}%
    #1{Ephesians}{Eph#2}%
    #1{Philippians}{Phil#2}%
    #1{Colossians}{Col#2}%
    #1{IThessalonians}{#5{1}Thess#2}%
    #1{IIThessalonians}{#5{2}Thess#2}%
    #1{ITimothy}{#5{1}Tim#2}%
    #1{IITimothy}{#5{2}Tim#2}%
    #1{Titus}{Tit#2}%
```

```
    #1{Philemon}{Philem#2}%
    #1{Hebrews}{Heb#2}%
    #1{James}{Jas#2}%
    #1{IPeter}{#5{1}Pet#2}%
    #1{IIPeter}{#5{2}Pet#2}%
    #1{IJohn}{#5{1}John}%
    #1{IIJohn}{#5{2}John}%
    #1{IIIJohn}{#5{3}John}%
    #1{Jude}{Jude}%
    #1{Revelation}{Rev#2}%
}
```

3.8.2 Number Styles

The book names can be customized with different number styles for book numbers and epistle numbers.

`\brm@number@arabic` The `\brm@number@arabic{`⟨*number*⟩`}` command formats a number with Arabic numerals.

```
\newcommand*{\brm@number@arabic}[1]{#1}
```

`\brm@number@lowerroman` The `\brm@number@lowerroman{`⟨*number*⟩`}` command formats a number with lower-case Roman numerals.

```
\newcommand*{\brm@number@lowerroman}[1]{\romannumeral#1}
```

`\brm@number@upperroman` The `\brm@number@upperroman{`⟨*number*⟩`}` command formats a number with upper-case Roman numerals.

```
\newcommand*{\brm@number@upperroman}[1]{%
    \MakeUppercase{\romannumeral#1}%
}
```

`\brm@number@arabicspace` The `\brm@number@arabicspace{`⟨*number*⟩`}` command formats a number with Arabic numerals and a trailing space.

```
\newcommand*{\brm@number@arabicspace}[1]{#1~}
```

`\brm@number@arabicspaceascii` The `\brm@number@arabicspaceascii{`⟨*number*⟩`}` command formats a number with Arabic numerals and a trailing ASCII-safe space, which is useful when defining a command name that can only consist of ASCII characters.

```
\newcommand*{\brm@number@arabicspaceascii}[1]{#1 }
```

`\brm@number@lowerromanspace` The `\brm@number@lowerromanspace{`⟨*number*⟩`}` command formats a number with lower-case Roman numerals and a trailing space.

```
\newcommand*{\brm@number@lowerromanspace}[1]{\romannumeral#1~}
```

`\brm@number@upperromanspace` The `\brm@number@upperromanspace{`⟨*number*⟩`}` command formats a number with upper-case Roman numerals and a trailing space.

```
\newcommand*{\brm@number@upperromanspace}[1]{%
    \MakeUppercase{\romannumeral#1}~%
}
```

\brm@number@scromanspace The \brm@number@upperromanspace{⟨*number*⟩} command formats a number with small caps Roman numerals and a trailing space.

```
\newcommand*{\brm@number@scromanspace}[1]{\textsc{\romannumeral#1}~}
```

\brm@number@upperromanascii The \brm@number@upperromanascii{⟨*number*⟩} command formats a number with upper-case Roman numerals, in an expandable ASCII-safe way, which is useful when defining a command name that can only consist of ASCII characters.

```
\newcommand*{\brm@number@upperromanascii}[1]{%
    \expandafter\brm@uppercaseascii\expandafter{\romannumeral#1}%
}
```

3.8.3 Book Aliases

Various book aliases are provided for all the book names, including versions of the full names, the abbreviated names, and the alternative abbreviated names. The book numbers can be done with either Arabic or Roman numerals. The abbreviations are supported both with and without periods. In order to support all such combinations, there are 10 different definitions. (If there were more than 24 combinations, then it would start to be necessary to programmatically keep track of them, but this small number can still be maintained manually).

```
\brm@foreachbook@full{\providebiblebookalias}{}{}{}{}{}{}%
    {\brm@number@upperromanascii}{\brm@number@upperromanascii}
\brm@foreachbook@full{\providebiblebookalias}{}{}{}{}{}{}%
    {\brm@number@arabic}{\brm@number@arabic}
\brm@foreachbook@abbrv{\providebiblebookalias}{}%
    {\brm@number@upperromanascii}{\brm@number@upperromanascii}
\brm@foreachbook@abbrv{\providebiblebookalias}{}%
    {\brm@number@arabic}{\brm@number@arabic}
\brm@foreachbook@abbrv{\providebiblebookalias}{.}%
    {\brm@number@upperromanascii}{\brm@number@upperromanascii}
\brm@foreachbook@abbrv{\providebiblebookalias}{.}%
    {\brm@number@arabic}{\brm@number@arabic}
\brm@foreachbook@altabbrv{\providebiblebookalias}{}{}%
    {\brm@number@arabic}{\brm@number@arabic}
\brm@foreachbook@altabbrv{\providebiblebookalias}{}{}%
    {\brm@number@upperromanascii}{\brm@number@upperromanascii}
\brm@foreachbook@altabbrv{\providebiblebookalias}{.}{}%
    {\brm@number@arabic}{\brm@number@arabic}
\brm@foreachbook@altabbrv{\providebiblebookalias}{.}{}%
    {\brm@number@upperromanascii}{\brm@number@upperromanascii}
```

3.8.4 Formatted Book Names

Several different forms of book names are created by default. These commands define the callback as \providebiblebook{⟨*stylename*⟩}, which creates a version of each book name in the given style.

```
\brm@foreachbook@full{\providebiblebook{fullname}}{\ }{}{}{}{}{}%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
```

```
\brm@foreachbook@abbrv{\providebiblebook{jerusalem}}{}%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
\brm@foreachbook@altabbrv{\providebiblebook{anglosaxon}}{.}{\ }%
    {\brm@number@upperromanspace}{\brm@number@upperromanspace}
\brm@foreachbook@altabbrv{\providebiblebook{JEH}}{.}{\ }%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
\brm@foreachbook@full{\providebiblebook{MHRA}}{\ }{}{}{}{}{}%
    {\brm@number@scromanspace}{\brm@number@scromanspace}
\brm@foreachbook@altabbrv{\providebiblebook{NTG}}{}{\ }%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
\brm@foreachbook@altabbrv{\providebiblebook{MLA}}{.}{\ }%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
\brm@foreachbook@altabbrv{\providebiblebook{chicago}}{.}{\ }%
    {\brm@number@arabicspace}{\brm@number@arabicspace}
```

3.8.5 Default Styles

All the styles that are supported by the bibleref package are supported here too. Each one is made to delegate to the `\standardbiblestyle` helper. The parameters to that helper determine the punctuation and spacing that will be used in the formatted output. The helper's signature is `\standardbiblestyle-{⟨bookstyle⟩}{⟨bkchsep⟩}{⟨chvsep⟩}{⟨bksep⟩}{⟨chsep⟩}{⟨vsep⟩}{⟨rangesep⟩}-{⟨chapterstyle⟩}{⟨versestyle⟩}{⟨beginpassage⟩}{⟨endpassage⟩}{⟨list⟩}`. The `\providebiblestyle` command creates an environment in which three parameters are defined, for the begin passage, the end passage, and the passage list.

New styles can be defined by creating new definitions like the ones here, except with different punctuation or spacing. Examples of how to define custom styles are found in the bibleref-lds package.

```
\providebiblestyle{fullname}{\standardbiblestyle{fullname}%
    {\ }{:}{; }{;}{,}{--}%
    {\brm@number@arabic}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{jerusalem}{\standardbiblestyle{jerusalem}%
    {\ }{:}{; }{; }{,}{--}%
    {\brm@number@arabic}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{anglosaxon}{\standardbiblestyle{anglosaxon}%
    {\ }{.}{; }{; }{,}{--}%
    {\brm@number@upperroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{JEH}{\standardbiblestyle{JEH}%
    {\ }{.~}{; }{; }{,}{--}%
    {\brm@number@lowerroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{MHRA}{\standardbiblestyle{MHRA}%
    {\ }{.~}{; }{; }{,}{--}%
    {\brm@number@lowerroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{NTG}{\standardbiblestyle{NTG}%
```

```
    {\ }{,}{; }{; }{,}{--}%
    {\brm@number@lowerroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{MLA}{\standardbiblestyle{MLA}%
    {\ }{.}{; }{; }{,}{--}%
    {\brm@number@lowerroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
\providebiblestyle{chicago}{\standardbiblestyle{chicago}%
    {\ }{\,:\,}{; }{; }{,}{--}%
    {\brm@number@lowerroman}{\brm@number@arabic}%
    {#1}{#2}{#3}}
```

3.9 Text Style

The style that outputs a textual description of the reference is more complicated than the others. It needs to be defined here in its own section.

This style depends on commands that are not expandable, so this style is unique in that it may not be used in an expandable context. It would be possible to implement this style in an expandable manner, by providing replacements for the fmtcount package's `\numberstringnum` and `\Ordinalstringnum` commands. The effort required would be large, however, and this style is less likely to be needed in an expandable context anyway.

`\brm@number@stomach@text` This special number format spells out the number using its English word. *This command relies on `\numberstringnum`, which is not expandable.*

```
\newcommand*{\brm@number@stomach@text}[1]{\numberstringnum{#1}}
```

`\brm@number@stomach@ordspace` This special number format spells out the number using its English ordinal word. This is useful for book numbers, such as `First`. *This command relies on `\Ordinalstringnum`, which is not expandable.*

```
\newcommand*{\brm@number@stomach@ordspace}[1]{\Ordinalstringnum{#1}~}
```

`\brm@number@stomach@chapter` This special number format prefixes the text with the label "chapter." It delegates to `\brm@number@stomach@text`. *This command relies on `\numberstringnum`, which is not expandable.*

```
\newcommand*{\brm@number@stomach@chapter}[1]{chapter\ %
    \brm@number@stomach@text{#1}}
```

`\brm@number@stomach@verse` This special number format prefixes the text with the label "verse." It delegates to `\brm@number@stomach@text`. *This command relies on `\numberstringnum`, which is not expandable.*

```
\newcommand*{\brm@number@stomach@verse}[1]{verse\ %
    \brm@number@stomach@text{#1}}
```

The style of book names spells out all the titles and uses English text instead of numerals.

```
\brm@foreachbook@full{\providebiblebook{text}}%
    {\ }%
```

```
    {Book of }{Gospel according to St.~}%
    {Epistle to }{Epistle to the }{Epistle of }%
    {\brm@number@stomach@ordspace}%
    {\brm@number@stomach@ordspace}
```

The style of references spells out all the conjunctions and numbers.

```
\providebiblestyle{text}{\standardbiblestyle{text}%
    {,\ }{\ }{,\ and\ }{,\ and\ }{,\ and\ }{\ to\ }%
    {\brm@number@stomach@chapter}{\brm@number@stomach@verse}%
    {#1}{#2}{#3}}
```

3.10 Bible Gateway Style

Since the parsing commands are all expandable, it is possible to parse a reference within a `\url` command. Then, the formatted output can be made to look like a valid URL rather than a human-readable style. So creating a link to a Bible reference uses all the same implementation machinery as the non-linked styles. The URL could be made to point to a variety of targets. This section illustrates how to create a link to Bible Gateway. The user may add other styles to link to any other sites.

First, the book names are defined. Bible Gateway uses the full names of the books and Arabic numerals. The ASCII version of the space character must be used, because a non-breaking space would not URL-encode to a space character correctly.

```
\brm@foreachbook@full{\providebiblebook{biblegateway.com}}%
    {\ }{}{}{}{}{}%
    {\brm@number@arabicspaceascii}{\brm@number@arabicspaceascii}
```

`\providebiblegatewayurl` The Bible Gateway site publishes many versions of the Bible, (e.g. NIV). A user that wishes to have their preferred version embedded in the URL should use `\providebiblegatewayurl{`⟨*urlstylename*⟩`}{`⟨*version*⟩`}`.

```
\newcommand*{\providebiblegatewayurl}[2]{%
    \providebiblestyle{#1}{%
        \brm@biblegatewayurl@style{#2}{##1}{##2}{##3}%
    }%
}
```

`\brm@biblegatewayurl@style` The Bible Gateway URL is constructed by the style `\brm@biblegatewayurl@style{`⟨*version*⟩`}{`⟨*beginpassage*⟩`}{`⟨*endpassage*⟩`}{`⟨*list*⟩`}`. Note that ampersands in the address are represented as `\&` instead of `&`, so it will still work in contexts where the ampersand is an active character. It uses the standard Bible style by calling `\brm@standard@filtered`, and it is allowed to specify a filter. The filter is `\brm@biblegatewayurl@filter`, which alters the standard format to meet the Bible Gateway requirements.

```
\newcommand*{\brm@biblegatewayurl@style}[4]{%
    http://www.biblegateway.com/passage/?search=%
    \brm@standard@filtered%
        {\brm@packseparators{ }{:}}%
```

```
        {\brm@packpassage{; }{; }{,}}%
        {\brm@packpassage{-}{-}{-}}%
        {\brm@packpassage%
            {\thebookname{biblegateway.com}}%
            {\brm@formatchapter{\brm@number@arabic}}%
            {\brm@formatverse{\brm@number@arabic}}}%
        {\brm@biblegatewayurl@filter{-}}%
        {#2}{#3}{#4}%
    \&version=#1%
}
```

`\brm@biblegatewayurl@filter` A filter for standard formats is `\brm@biblegatewayurl@filter{`⟨*rangesep*⟩`}{`⟨*lastpassage*⟩`}{`⟨*passage*⟩`}{`⟨*infixseparators*⟩`}{`⟨*prefixes*⟩`}{`⟨*formatters*⟩`}{`⟨*delegate*⟩`}`. It makes sure that each new passage in a list contains the book name, not just the chapter or verse. This is because Bible Gateway sometimes behaves unexpectedly if a number is specified without a book name, and it is ambiguous whether the number represents a chapter or a verse. Ranges, (as opposed to lists), behave normally and are not altered.

```
\newcommand*{\brm@biblegatewayurl@filter}[7]{%
    \brm@ifsamestr{\theverse{#5}}{#1}{%
        % Show ranges like normal.
        #7{#2}{#3}{#4}{#5}{#6}%
    }{%
        % List the book explicitly for each list element.
        #7{\brm@packpassage{}{}{}}{#3}{#4}{#5}{#6}%
    }%
}
```

A URL style that creates links for Bible Gateway references is defined. Users may define other styles for different versions of the Bible at Bible Gateway.

```
\providebiblegatewayurl{biblegateway.com-url}{NIV}
```

`\providebiblegatewaystyle` Different users might wish to define links for a variety of styles. For example, a user might wish to have either full text or abbreviated text being linked. To provide this flexibility, there is the `\providebiblegatewaystyle{`⟨*stylename*⟩`}{`⟨*urlstylename*⟩`}{`⟨*textstylename*⟩`}` command. The user first gives the name for this new style, and then gives the name of the existing style that will be used to format the link, and then finally the name of the existing style that will be used to format the linked text.

```
\newcommand*{\providebiblegatewaystyle}[3]{%
    \providebiblestyle{#1}{%
        \href{%
            \brm@delegatestyle{#2}{##1}{##2}{##3}%
        }{%
            \brm@delegatestyle{#3}{##1}{##2}{##3}%
        }%
    }%
}
```

A style is defined that uses a link style for the Bible Gatway NIV version and full names for the linked text. Users may define other styles for different link formats or different linked text formats.

```
\providebiblegatewaystyle{biblegateway.com}%
    {biblegateway.com-url}{fullname}
```